Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Francisco Hoyos Garcia

Automatisierte Ansteuerung von gekoppelten Systemmodellen in der Flugzeugentwicklung

Fakultät Technik und Informatik
Department Maschinenbau und Produktion

Faculty of Engineering and Computer Science
Department of Mechanical Engineering and Production Management

Francisco Hoyos Garcia

Automatisierte Ansteuerung von gekoppelten Systemmodellen in der Flugzeugentwicklung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Entwicklung und Konstruktion
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
Centerline Design GmbH
Hein-Saß-Weg 36
21129 Hamburg

Erstprüfer/in: Prof. Dr. Jutta Abulawi
Zweitprüfer/in : Robert Sowarka

Abgabedatum: 26.04.2024

Zusammenfassung

Name des Studierenden

Francisco Hoyos Garcia

Thema der Bachelorthesis

Automatisierte Ansteuerung von gekoppelten Systemmodellen in der Flugzeugentwicklung

Stichworte

Automatisierte Ansteuerung, Magic Cyber System Engineer, Python, UML, XML, SysML, GUI.

Kurzzusammenfassung

Diese Arbeit umfasst den systematischen Einsatz von System Engineering für die Erstellung von Anforderungen, Use Cases, die für die Entwicklung einer grafischen Benutzeroberfläche verwendet werden. Diese Anwendung soll Systemmodelle steuern, die die Vordimensionierung eines Flugzeugs erstellen, indem der Benutzer die Startparameter eingibt und eine oder mehrere Simulationen startet.

Name of Student

Francisco Hoyos Garcia

Title of the paper

Automated control of coupled system models in aircraft development

Keywords

Automated control, Magic Cyber System Engineer, Python, UML, XML, SysML, GUI.

Abstract

This work involves the systematic use of systems engineering for the creation of requirements, use cases that are used for the development of a graphical user interface. This application is intended to control system models that create the pre-dimensioning of an aircraft by the user entering the start parameters and starting one or more simulations.

Inhaltsverzeichnis

Abkürzungsverzeichnis

APU	Auxilary Power Unit
API	Application Programming Interface
DLR	Deutsches Zentrum für Luft- und Raumfahrt
EU	Europäische Union
GATE	Green Aviation Technologies
GUI	Graphical User Interface
HAW	Hochschule für Angewandte Wissenschaften Hamburg
MBSE	Model Based Systems Engineering MIWa Forschungsprojekt
MCSE	Magic Cyber System Engineer
OMG	Object Management Group
OOP	Objektorientierte Programmierung
SysML	Systems Modeling Language
UML	Unified Modeling Language
W3C	World Wide Web Consortium

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1 Hintergrund

Eine der großen Herausforderungen für die Luftfahrtindustrie ist die Notwendigkeit, den Betrieb von Luftverkehrssystemen immer nachhaltiger und klimaneutraler zu machen. Es müssen Maßnahmen getroffen werden, um den Klimaeinfluss durch die Einführung von alternativen zu fossilen Brennstoffen zu verringern [11]. Wasserstoff ist eine vielversprechende Alternative zum direkten Ersatz oder zumindest zur Reduzierung des Einsatzes von Kerosin. Dieses ist auf der Erde weit verbreitet und frei von natürlichen Ressourcen wie Erdöl und wirtschaftlichen Ungleichgewichten [4]. Darüber hinaus ist die spezifische Energie von Wasserstoff, die als Energie in Gewichtseinheiten ([J/kg]) gemessen wird, höher als die von Kerosin. Dies bedeutet, dass bei gleicher Brennstoffmenge eine höhere Energiemenge zur Verfügung steht [28, S.116–117].

Für die Bewertung, Entwicklung und Umsetzung neuer Innovationen kann es von Vorteil sein, moderne Methoden und Systemtechnologien für die Entwicklung einzusetzen. Eines der Beispiele ist die Entwicklungsmethodik Model-Based Systems Engineering (MBSE) (s. Kapitel 2.1). MBSE wird eingesetzt, um bei der Entwicklung komplexer Systeme den Systemumfang nicht aus den Augen zu verlieren, den Kontext im Auge zu behalten und Anforderungen einfach vergleichen zu können. Aus diesem Grund wird diese Methode zunehmend bei komplexen Systemen wie Flugzeugen eingesetzt.

Diese Aufgabe haben sich der mittelständischer Ingenieursdienstleister, Centerline Design GmbH, das Deutsches Zentrum für Luft- Raumfart (DLR) und die Hochschule für Angewandte Wissenschaften (HAW Hamburg) zum Ziel gesetzt. Das kooperations- und Forschungsprojekt MIWa (s. Kapitel 1.2) ist dafür da verschiedene Flugzeugskonfiguration zu vergleichen.

1.2 Das Forschungsprojekt MIWa

Das Hamburger Förderprogramm Green Aviation Technologies (GATE) unterstützt gezielt Luftfahrtprojekte, die das Fliegen nachhaltiger machen sollen. Das Förderprogramm wurde von Hamburg Aviation im Rahmen der Initiative „Technology Roadmap“ entwickelt und unterstützt. Die Finanzierung von GATE übernimmt die IFB Hamburg im Auftrag der Stadt Hamburg [10]. Eines der Forschungsprojekte ist MIWa (MBSE-basierte Integration & Variantenbildung von Wasserstoffkryodrucktanksystemen zukünftiger Flugzeugkonfigurationen).

Das grundlegende Ziel des Projektes MIWa besteht in der Entwicklung eines digitalen Systemmodells, das die Integration von Wasserstoffsystemen in ein Flugzeug und die Auslegung und das Benchmarking verschiedener Varianten ermöglicht. Hierbei werden die Anforderungen, Systemeigenschaften und Abhängigkeiten modelliert und die erstellten Datenmodelle mit 3D-Geometriemodellen sowie verschiedene Modelle verknüpft. Zur

vollständigen Digitalisierung wird auf Grundlage des Systemmodells eine virtuelle Umgebung geschaffen, in der diverse zu entwickelnde Kabinenlayouts und Systeme, sowie das in den Flugzeugrumpf zu integrierende Wasserstofftanksystem, visualisiert werden können. Dies ermöglicht die Analyse und Bewertung geometrischer Aspekte von Designvarianten in einer virtuellen Umgebung [8].

1.3 Aufgabenstellung

In dieser Bachelorarbeit soll eine Softwareanwendung für die automatisierte Ansteuerung der gekoppelten Systemmodelle entwickelt werden. Der Fokus liegt in der Gestaltung einer ergonomischen Benutzeroberfläche (GUI) bei gleichzeitiger Modularität und flexibler Rekonfigurierbarkeit des Gesamtsystemmodells. Die Entwicklung der GUI beinhaltet die Identifikation und Analyse der Stakeholder sowie die Definition von Personas und die Ermittlung von zugehörigen User Stories, um die Anforderungen an die GUI zu präzisieren. Hierauf aufbauend sollen in der
Systemmodellierungssprache SysML verschiedene Use Cases definiert werden. Die Use Cases sind hinsichtlich ihrer Relevanz, ihrer Komplexität und ihrer Realisierbarkeit zu bewerten. Für einen ausgewählten Use Case soll eine parametrisierte GUI mit der Programmiersprache Python entwickelt werden, die automatisch auf verschiedene vorhandene Partialmodelle zugreift und Systementwickler bei der Beurteilung von Systemarchitekturvarianten unterstützt.

Auflistung der geplanten Arbeitsschritte:

- Einarbeitung in die im Projekt MIWa entwickelten Partialmodelle und deren vorbereitete Schnittstellen für den externen Zugriff (Input und Output)
- Einarbeitung in die methodische Definition von Personas und Use-Cases
- Priorisierung und Auswahl eines Use-Case
- Entwicklung einer GUI für den gewählten Use-Case in Python
- Umsetzung der automatisierten Modellansteuerung
- Umsetzung der User Stories
- Definition von Test-Cases zur Verifizierung und Validierung
- Durchführung der verifizierenden Test-Cases
- Durchführung von validierenden Anwendertests
- Auswertung der Ergebnisse
- Diskussion der Ergebnisse
- Dokumentation in einer wissenschaftlichen Arbeit (einschließlich des Programm-Codes, der genutzten Modelle und aller Simulations- und Testergebnisse)

Diese Abschlussarbeit wird in Zusammenarbeit mit der Centerline Design GmbH durchgeführt.

2 Stand der Technik

Im folgenden Kapitel werden die für diese Arbeit relevanten Begriffe erläutert.

2.1 Model Based Systems Engineering

Model-Based Systems Engineering (MBSE) ist eine leistungsstarke Technik beim Entwurf komplexer Systeme. Ein Beispiel dafür ist die teilweise Anwendung von MBSE-Programmen bei der Entwicklung der nächsten Generation von Passagierflugzeugen. Ähnlich wie in anderen Projekten gibt es weitere Vorhaben in verschiedenen Branchen und Fachdisziplinen, wie zum Beispiel im Maschinenbau, der Luft- und Raumfahrt sowie der Automobilindustrie, die die Vorteile von MBSE aufzeigen und nutzen [32].

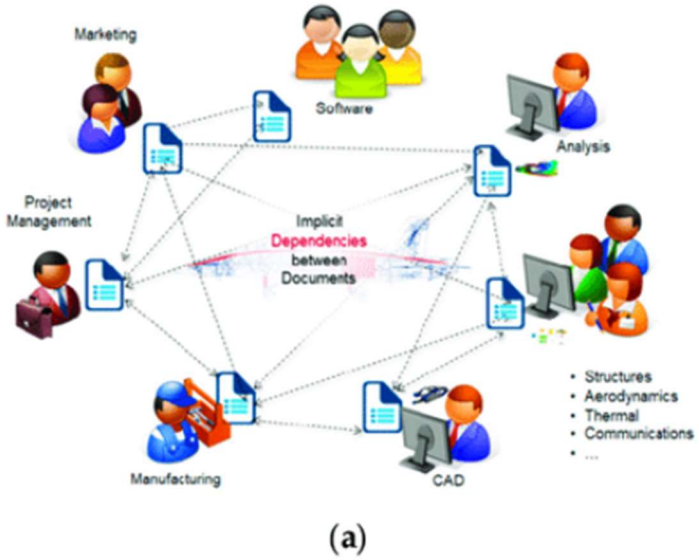


Abbildung 1: Klassische Entwicklungsmethoden (von R. Uppal, 2022, Idstch).

Die Vorteile von MBSE umfassen u.A. die Verbindung der Ergebnisse mit den Systemspezifikationen für die Analyse, die Verbesserung der Kommunikation zwischen Modellierern und Stakeholdern und die Verbesserung des Wissenserwerbs [12]. Aus Abbildung 2 geht hervor, dass bei der klassischen Methode, wie der traditionellen Dokumentation, die Kommunikationsmittel zwischen den verschiedenen Abteilungen, Mitarbeitern oder Disziplinen nicht zentralisiert sind. Dies kann zu umfangreichen Dokumentationslisten führen, um Änderungen im Gesamtsystem zu verstehen, ohne die Auswirkungen einer Änderung in einem Teil des Systems angemessen zu berücksichtigen. Die Anwendung einer zentralisierten und neutralen Methode, wie in Abbildung 2 dargestellt, die nicht an eine bestimmte Disziplin gebunden ist (MBSE), ermöglicht jedoch eine direktere Erkennung von Änderungen auf globaler Ebene, die durch ein Teilsystem verursacht werden können.

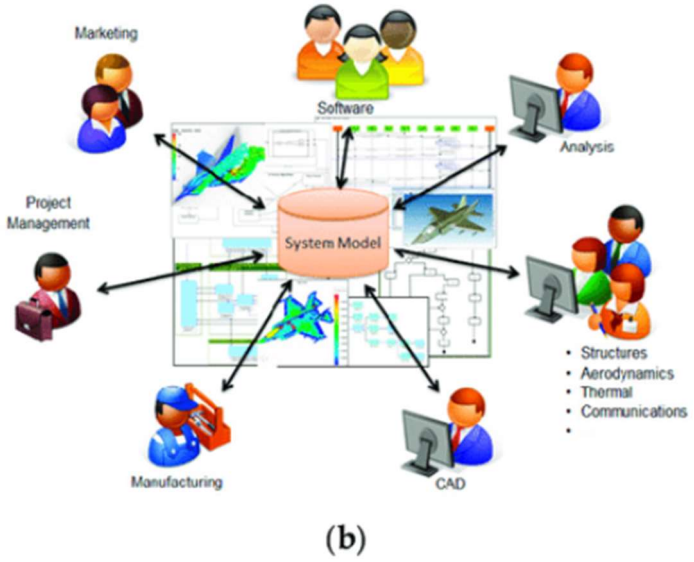


Abbildung 2: MBSE-Methode (von R. Uppal, 2022, Idstch).

In den Diskussionen über MBSE wird die Terminologie in Bezug auf die drei Fundamentsäulen verwendet, die diese Disziplin konstituieren. Die Definitionen dieser Säulen sind in der Abbildung 3 dargestellt [33]:

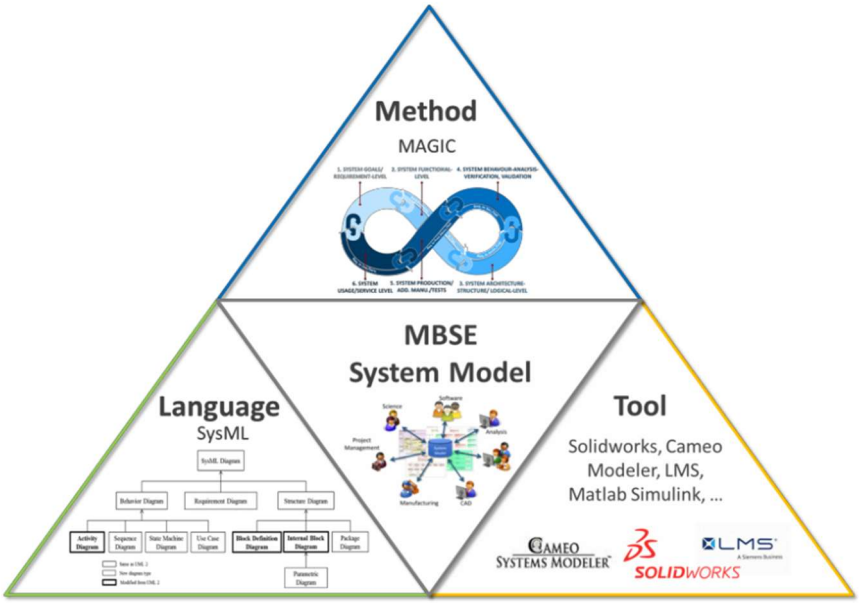


Abbildung 3: Dastellung der drei Grundlagen von MBSE (von R. Uppal, 2022. Idstch).

Die erste Säule bezieht sich auf die Systementwicklungssprache. Darauf folgt als zweite Säule das zugehörige Werkzeug. Schließlich wird die Modellierungsmethode als letzte Säule der Systemmodellierung betrachtet, wobei sich MIWa im Vergleich zu anderen Projekten, die MBSE aktiv einsetzen, durch die Verwendung einer anderen Methode auszeichnet. Im Folgenden werden die drei Säulen näher erläutert [33]:

2.1.1 Systementwicklungssprache

Für die Erstellung von Modellen im MBSE wird in diesem Projekt die Systems Modeling Language (SysML) verwendet. Dabei handelt es sich um eine Variante der Unified Modeling Language (UML). UML ist eine Softwareentwicklungssprache, die darauf abzielt, die Entwicklung komplexer Software zu erleichtern und den Überblick über die Anforderungen komplexer Programme zu behalten. In den 1990er Jahren wurde die erste Version der UML veröffentlicht, die von verschiedenen Entwicklern, darunter Grady Booch, James Rumbaugh und Ivar Jacobson, als Notation und Diagrammdarstellung für die Software-Dokumentation entwickelt wurde. Derzeit ist die Object Management Group (OMG) für die Weiterentwicklung der Sprache zuständig, deren letzte erfolgreiche Version UML 2.5.1 im Jahr 2017 veröffentlicht wurde [35, S. 201–205].

Die Notwendigkeit einer Erweiterung der UML für die allgemeine Systemmodellierung führte zur Entwicklung der SysML durch die SysML Partner Working Group. Diese Gruppe besteht aus verschiedenen Vertretern von Unternehmen und Organisationen, darunter Lockheed Martin Corporation, NASA, IBM und INCOSE. Im Jahr 2003 wurde der erste Entwurf der Sprache erstellt und derzeit wird an Version 2.0 der SysML gearbeitet [35, S. 309–312]. Für die Entwicklung der Modelle im MIWa-Projekt wird die zuletzt veröffentlichte SysML-Version 1.6 verwendet. Diese Version wird als Standardsprache von der Modellierungssoftware Magic Cyber Systems Engineer verwendet.

Auch bei der Entwicklung der in dieser Abschlussarbeit beschriebenen Anwendung wird SysML konsequent eingesetzt, um den Bedürfnissen der Projektbeteiligten gerecht zu werden. Die Vorgehensweise folgt den praktischen Methoden, die Tim Weichen in „Systems Engineering mit SysML und UML“ beschreibt (siehe Kapitel 2.1).

2.1.2 Das Werkzeug

Magic Cyber Systems Engineer ist das Hauptprogramm in dem Forschungsprojekt MIWa, da Systemingenieure hiermit Modelle entwickeln, simulieren und validieren können. Das Programm wurde hauptsächlich von No Magic. Inc. Entwickelt und dann von Dassaut Systems gekauft und weiterentwickelt. Das Kooperationsprojekt arbeitet mit der Version 2022x R1 HF1.

Die Applikation bietet ein innovatives Modellausführungs-Framework. Dieses Framework ist erweiterbar und basiert auf anerkannten Standards der Object Management Group (OMG) für ausführbares UML (fUML) und des World Wide Web Consortiums (W3C) für State Chart XML (SCXML). Mit der Magic Model Analyst-Erweiterung des UPDM-Plugins kann man das Verhalten von Systemen simulieren. Diese Validierung erfolgt durch Ausführen und Debuggen parametrischer SysML-Modelle. Dies geschieht vor dem Hintergrund eines stark vereinfachten Modells der geplanten Benutzerschnittstelle [18].

Diese Software ermöglicht den Benutzern die Reaktion des Systems auf Benutzerinteraktionen und vordefinierte Testdaten und Ausführungsszenarien zu bewerten. Dieser Prozess entspricht dem Zweck der Simulation, der darin besteht, das System zu verstehen, ohne in das reale System einzugreifen. Dieser Verzicht auf Eingriffe kann darauf zurückgeführt werden, dass das reale System nicht definiert oder nicht verfügbar ist oder das Kosten-, Zeit-, Ressourcen- oder Risikobeschränkungen bestehen. Simulationen werden normalerweise anhand eines Modells des Systems durchgeführt [19].

2.1.3 Die Modellierungsmethode

Eine Modellierungsmethodik definiert die aufeinander folgenden Operationen, die bei der Erstellung einzelner Modelle durchgeführt werden. Der Ablauf der meisten Methoden im Kontext des MBSE orientiert sich an den Schritten Anforderungsdefinition, Entwurf, Analyse sowie Verifikation und Validierung. Entscheidend ist dabei, dass die Modellierungsmethode eindeutig festlegt, welche Informationen zu welchem Zeitpunkt in welchem Detaillierungsgrad benötigt werden [29].

Die im Rahmen des Projekts gewählte Modellierungsstrategie sieht eine Unterteilung des Gesamtsystems Modell in mehrere Teilmodelle vor. Ein Teilmodell umfasst ein SysML-Modell für die Architekturmodellierung und, falls erforderlich, zusätzliche Modelle. Eine der Vorteile dieser Entscheidung liegt in der Modularität der Teilmodelle, wodurch die Wartbarkeit vereinfacht wird. Dies ergibt sich daraus, dass Submodelle erweitert oder ausgetauscht werden können. Dies eröffnet die Möglichkeit, unterschiedliche Variationen und Ergebnisse zu erzielen, beispielsweise über den Austausch eines Kerosintriebwerks durch ein Wasserstofftriebwerk. Dies führt zu einem Vergleich verschiedener Systeme mit dem Ziel, eine optimale Konfiguration zu definieren. Ein weiterer Vorteil, der im Projekt gewählten Modellierungsmethode, ist die interne Strukturierung. Die interne Strukturierung und Organisation der SysML-Modelle innerhalb der Entwicklungsplattform erfolgen nach dem EVA-Prinzip. Das EVA-Prinzip steht für „Eingabe, Verarbeitung, Ausgabe“ und beschreibt einen grundlegenden Verlauf in der Informatik und Informationsverarbeitung. Die Ursprünge dieses Prinzips können auf die Grundlagen der Informationstheorie und der Computertechnologie zurückverfolgt werden [8].

2.2 XML

Das XML-Format (Extensible Markup Language) wurde vom SGML (Standard Generalized Markup Language) Editor's Review Board entwickelt, das 1996 unter der Schirmherrschaft des World Wide Web Consortium (W3C) gegründet wurde [7]. XML ist keine Programmiersprache, sondern eine Inhaltsbeschreibungssprache. Das bedeutet, dass diese Datenstruktur keine ausführbare Funktion hat und nicht wie ein in C++ oder einer anderen Programmiersprache geschriebenes Programm kompiliert werden kann. Stattdessen bietet sie die Möglichkeit, Informationen strukturiert zu speichern und erlaubt eine erweiterte Anzahl von Tags, was bspw. in HTML nicht ohne Mehraufwand möglich ist [24].

Die Strukturierung und Hierarchisierung von Informationen wird als optimaler Ansatz für den Datentransfer zwischen Modellen auf unterschiedlichen Plattformen angesehen. Dies ist auf die menschliche Lesbarkeit und die hohe Kompatibilität mit den in diesem Forschungsprojekt verwendeten Programmen wie MATLAB, Python (siehe Kapitel 2.3), Catia V5 und Magic Cyber Systems Engineer zurückzuführen. In Abbildung 4 ist die hierarchische Struktur von XML-Dateien zu erkennen. Jede Bezeichnung, wie zum Beispiel 'MIWa', muss mit den Zeichen '<' und '>' umschlossen werden. Zusätzliche Informationen, wie beispielsweise 'version', werden als untergeordnete Bezeichnungen von 'MIWa' erkannt und können auch einen Wert haben. Zusätzliche Eigenschaften eines Tags, wie z.B. die Versionsnummer, sind für die Korrektheit des XML nicht zwingend erforderlich. Letztendlich müssen jedoch alle Tags korrekt mit '</' und '>' abgeschlossen werden.

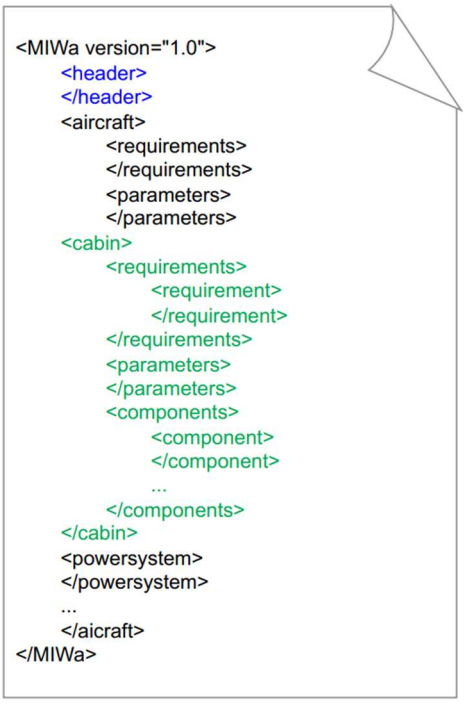


Abbildung 4: Exemplarische Struktur eine XML-Datei in dem Projekt (von M. Fuchs, 2023, DLR).

2.3 Python

Die offizielle Python-Website, auf der verschiedene Versionen heruntergeladen werden können und Änderungen in jeder Version dokumentiert sind, bietet eine umfassende Definition der Programmiersprache Python: *„Python ist eine interpretierte, interaktive und objektorientierte Programmiersprache. Sie enthält Module, Ausnahmen, dynamische Typisierung, dynamische Datentypen auf sehr hohem Niveau und Klassen. Sie unterstützt mehrere Programmierparadigmen, die über die objektorientierte Programmierung hinausgehen, wie die prozedurale und die funktionale Programmierung. Python verbindet eine bemerkenswerte Leistungsfähigkeit mit einer sehr klaren Syntax. Es verfügt über Schnittstellen zu zahlreichen Systemaufrufen und Bibliotheken sowie zu verschiedenen Fenstersystemen und kann in C oder C++ erweitert werden.“* [9].

Im Jahr 1989 wurde Guido van Rossum aufgrund seiner Unzufriedenheit mit der Programmiersprache ABC inspiriert, eine neue Sprache zu entwickeln. Er war insbesondere unzufrieden mit der eingeschränkten Erweiterbarkeit der Funktionalität, der unrealistischen Vorstellung, diese Probleme zu lösen, und der Notwendigkeit, die Systeme auf andere Art und Weise als die der Amoeba-Gruppe zu administrieren. Das Ergebnis war Python [9].

Python ist eine objektorientierte Programmiersprache (OOP). Dabei handelt es sich um ein Programmierparadigma, das auf der Konzeptualisierung von Software durch die Modellierung realer oder abstrakter Objekte beruht. Die Objektorientierung ist eine bewährte Methode zur Beherrschung komplexer Softwaresysteme. Dadurch wird der Code besser strukturiert, wiederverwendbar und leichter wartbar. Die klare Trennung der Verantwortlichkeiten in den einzelnen Objekten erleichtert das Verständnis und die Entwicklung komplexer Systeme. Darüber hinaus fördert sie die Erweiterbarkeit und Anpassbarkeit der Software [16, S. 30-32]. Ein weiteres Merkmal der objektorientierten Programmierung (OOP) ist die Klassenvererbung, die eine Klasse befähigt, die Eigenschaften und Funktionen einer externen Klasse zu übernehmen und zusätzliche Daten in ein Objekt einzufügen [1]. Dies eignet sich ideal für die Entwicklung komplexer Projekte, wie beispielsweise einer Desktop-Anwendung mit einer GUI. Durch die Verwendung von verschiedenen Klassen und Objekten wird die Modularität und Übersichtlichkeit des gesamten Projekts verbessert.

Python-Module sind externe Skripte in Python, die von Dritten entwickelt wurden. Sie können mit geringem Aufwand in das eigene Skript importiert werden und erweitern die Funktionalität des Programms [34]. Folgend werden die wichtigsten Module beschrieben, die im Zuge dieser Arbeit genutzt wurden.

xml.etree.ElementTree: *„Das Modul xml.etree.ElementTree implementiert eine einfache und effiziente Application Programming Interface (API) zum Parsen und Erstellen von XML-Daten.“* [34].

tkinter: *„ Das tkinter-Paket ist die Standard-Python-Schnittstelle zum Tcl/Tk-GUI-Toolkit. Es ist auf den meisten Unix-Plattformen, einschließlich macOS, sowie auf Windows-Systemen verfügbar.”* [27].

ttkbootstrap: *„Eine leistungsstarke Theme-Erweiterung für Tkinter, die von Bootstrap inspirierte On-Demand-Themes im modernen Flat-Stil ermöglicht.“* [30].

pandas: *„Pandas ist ein Python-Paket, das schnelle, flexible und ausdrucksstarke Datenstrukturen bereitstellt, die die Arbeit mit „relationalen“ oder „beschrifteten“ Daten sowohl einfach als auch intuitiv machen sollen. Es soll der grundlegende High-Level-Baustein für die Durchführung praktischer, realer Datenanalysen in Python sein. Darüber hinaus verfolgt es das umfassendere Ziel, das leistungsstärkste und flexibelste Open-Source-Tool zur Datenanalyse/-manipulation zu werden, das in jeder Sprache verfügbar ist. Es ist bereits auf einem guten Weg, dieses Ziel zu erreichen.“* [24].

json: *„Das Modul ist ein leichtes Datenaustauschformat, das von der JavaScript-Objektliteralsyntax inspiriert ist “* [14].

difflib: *„Dieses Modul stellt Klassen und Funktionen zum Vergleichen von Sequenzen bereit. Es kann beispielsweise zum Vergleichen von Dateien verwendet werden und kann Informationen über Dateiunterschiede in verschiedenen Formaten, einschließlich HTML, Kontext und Unified Diffs, erzeugen“* [5].

time: *„Dieses Modul stellt verschiedene zeitbezogene Funktionen zur Verfügung.“* [26].

PIL: “*Die Python Imaging Library erweitert Ihren Python-Interpreter um Bildverarbeitungsfunktionen. Diese Bibliothek bietet umfassende Dateiformatunterstützung, eine effiziente interne Darstellung und ziemlich leistungsstarke Bildverarbeitungsfunktionen. Die Kernbildbibliothek ist für den schnellen Zugriff auf Daten konzipiert, die in einigen grundlegenden Pixelformaten gespeichert sind. Es sollte eine solide Grundlage für ein allgemeines Bildverarbeitungstool bieten.“* [23].

os: *„Dieses Modul bietet eine portable Möglichkeit zur Nutzung betriebssystemabhängiger Funktionen.“* [23].

Pyautogui: *„PyAutoGUI ermöglicht es Python-Skripten, die Maus und Tastatur zu steuern, um die Interaktion mit anderen Anwendungen zu automatisieren. Die API ist einfach gehalten und funktioniert unter Windows, macOS und Linux mit Python 2 und 3.“* [36].

In diesem Projekt wird die Version 3.11 von Python mit Hilfe des Anacona-Modulmanagers verwendet. Anaconda bietet den Vorteil, Python Virtual Environments zwischen Computern zu teilen. Dadurch ist es möglich, auf verschiedenen Rechnern mit der gleichen Python-Version und den gleichen Modulen zu arbeiten.

2.4 Automatisierung

Nach dem technischen Wörterbuch, DIN IEC 60050, ist automatisch wie folgt definiert: *„einen Prozess oder eine Einrichtung bezeichnend, der oder die unter festgelegten Bedingungen ohne menschliches Eingreifen abläuft oder arbeitet.“* Softwareanwendungen, die üblicherweise von Ingenieuren verwendet werden, wie z. B. Excel oder CATIA, können durch ihre Makroprogrammierfähigkeiten, bspw. mit Visual Basic, automatisiert werden. Dies ist ein

gutes Beispiel für die Automatisierung in der Softwareentwicklung. Dies ist das Ziel dieser Abschlussarbeit: Die Teilautomatisierung von MCSE, um die Iteration und Validierung von verschiedenen Flugzeugkonfigurationen, das Benchmarkings der Simulationsergebnisse und zugehörigen Anforderungen zu vergleichen. Dies wird durch eine XML-Brücke zwischen den Modellen ermöglicht, wobei Magic Cyber Systems Engineer (siehe Kapitel 2.1) selbst und Python (siehe Kapitel 2.3) für den Austausch zwischen den Modellen genutzt werden. Der Informationsfluss aus der Simulation wird durch die Verwendung von XML-Dateien (siehe Kapitel 2.2) unterstützt.

2.5 GUI & Ergonomie

Die Gestaltung der grafischen Benutzeroberfläche (GUI) ist von wesentlicher Bedeutung bei der Entwicklung von Programmen, da diese eine visuelle Verbindung zwischen Benutzer und Maschine darstellt. Das Design definiert nicht nur die Bedienung der Software, sondern auch die Wahrnehmung des Benutzers. Ein gut durchdachtes GUI-Design verwandelt Algorithmen in Anwendungen und verleiht ergonomisches Aussehen [37, S. 17-19].

Die Benutzerfreundlichkeit, der Anwendung, die Zuverlässigkeit des Programms und das Design sind entscheidende Faktoren, die darüber entscheiden, ob ein Programm wirklich nützlich ist. Ein Erschweren der durchzuführenden Aufgaben durch die GUI können dazu führen, dass die Softwareanwendung vom Nutzer nicht verwendet wird.

2.6 Künstliche Intelligenz (KI)

Künstliche Intelligenz (KI) hat als Schlüsseltechnologie das Potenzial, ganze Märkte, Branchen, Geschäftsaktivitäten und Geschäftsmodelle grundlegend zu verändern [33]. Das erste freie Sprachmodell, das Ende 2022 veröffentlicht wurde, heißt „GPT-3“ von OpenAI und ist gleichzeitig das bekannteste. Diese spezielle Technologie basiert auf Benutzereingaben in Form von Fragen, Vorschlägen oder einfachem Text. Hier ein praktisches Beispiel, wie sich GPT-3 mit seiner Frontend-Maske ChatGPT selbst beschreibt:

„ChatGPT ist ein KI-gestütztes Sprachmodell, das von OpenAI entwickelt wurde. Es basiert auf der GPT-3.5-Architektur (Generative Pre-trained Transformer 3.5). "GPT" steht für "Generative Pre-trained Transformer", und es handelt sich um ein Modell, das darauf trainiert wurde, menschenähnlichen Text zu generieren, basierend auf den Mustern und Informationen, die es aus einer großen Menge von Textdaten gelernt hat.

ChatGPT kann für eine Vielzahl von natürlichen Sprachverarbeitungsaufgaben verwendet werden, einschließlich Konversationen, Textgenerierung, Fragen beantworten und vieles mehr. Es kann aufgrund seines prägenden Trainings mit einer breiten Palette von Anfragen und Themen umgehen. Es ist wichtig zu beachten, dass meine Informationen bis Januar 2022 aktuell sind, und es könnte Fortschritte oder neue Versionen von ChatGPT geben, die nach diesem Datum entwickelt wurden.“ [3].

Eine Fähigkeiten dieser künstlichen Intelligenz umfassen die Generierung von funktionalem Code in verschiedenen Programmiersprachen wie C++, Java oder Python. Diese Funktionalität erweist sich als äußerst hilfreich bei der Identifizierung spezifischer Lösungen, die sonst meist nur über Internetforen zu finden sind, und beschleunigt die Umsetzung praktischer Lösungen, auch wenn sie in den meisten Fällen nicht die effizienteste ist. In bestimmten Abschnitten und Problemen dieser Arbeit wurden diese Werkzeuge eingesetzt, um den Programmierer bei der Bewältigung von Herausforderungen zu unterstützen, wie z.B. beim Auffinden unbekannter Bibliotheken oder Methoden und bei der Generierung von Lösungen für Probleme, für die das Fachwissen des Programmierers allein nicht ausreichte. Die verbreitete Verwendung dieses Werkzeugs hat nach dem heutigen Stand noch nicht dazu geführt, dass komplexer Code in der Praxis fehlerfrei generiert werden kann.

2.7 Betriebssystem

Ein Betriebssystem ist eine Softwarekomponente, die auf einem Computer oder einem Informationssystem läuft und die Ressourcen verwaltet. Es fungiert als Schnittstelle zwischen der Hardware und den Anwendungsprogrammen. Das Betriebssystem spielt eine entscheidende Rolle bei der Verwaltung von Prozessen, dem Zugriff auf Speicher, der Dateiverwaltung und der Kommunikation zwischen Hardwarekomponenten. Es dient als essenzielle Grundlage für die Ausführung von Anwendungen und Programmen [2].

Für diese Anwendung wurde Windows 10 verwendet. Windows 10 ist ein Betriebssystem, das von der Microsoft Corporation entwickelt wurde und im Jahr 2015 auf den Markt gebracht wurde [25, S. 17–56].

2.8 Code-Editor

Ein Editor ist eine Softwareanwendung, die speziell für das Schreiben und Bearbeiten von Dateien entwickelt wurde. Code-Editoren sind textbasiert und bieten Funktionen wie Syntaxhervorhebung und Code-Faltung [2]. Der vom Entwickler empfohlene Code-Editor ist Visual Studio Code (VSCode). Er ist ein kostenloser, quelloffener Code-Editor von Microsoft. Der Editor unterstützt eine Vielzahl von Programmiersprachen und bietet eine Architektur, die es Entwicklern ermöglicht, Funktionalitäten durch zahlreiche verfügbare Erweiterungen hinzuzufügen. Visual Studio Code hebt sich durch seine Schnelligkeit, Benutzerfreundlichkeit und Vielseitigkeit hervor [15].

3 Stand des gesamten Projekts

Eine Herausforderung bei der Entwicklung von wasserstoffbasierten Energieversorgungssystemen in der zivilen Luftfahrt ist die geometrische und technische Integration in das Verkehrsflugzeug sowie die Wechselwirkungen mit anderen Systemen. Aufgrund der derzeitigen Unklarheit über die optimale Systemarchitektur und Integrationsform, die zu einer optimalen Gesamtkonstruktion des Flugzeugs führen kann, ist eine umfassende Untersuchung zahlreicher Konfigurationsmöglichkeiten erforderlich. Daher kann die Modellierungsmethodik des Projekts (siehe Kapitel 2.1) am angewendet werden, um die verschiedenen Varianten dieses Flugzeugs zu untersuchen.

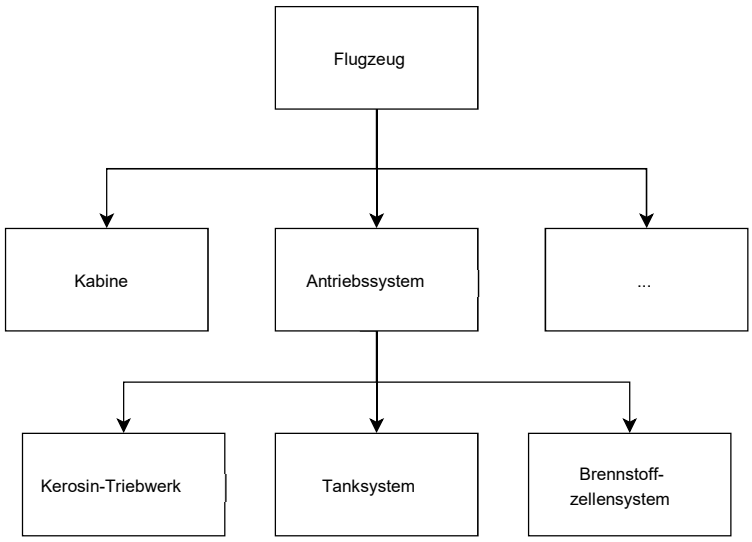


Abbildung 5: Erste Variante eines Flugzeuges durch MIWa. (von M. Fuchs, 2023, DLR).

Mit Hilfe der in Abbildung 5 dargestellten Systemhierarchie ist es möglich, eine Flugzeugvarianten und die Modularität der Systeme zu verstehen. Der Fokus liegt darauf, wie sich Änderungen in den Subsystemen eines vorhandenen Flugzeugs, das Airbus A320 Neo, sich auswirken. Dies betrifft beispielsweise die Anzahl der Passagiere. Das gesamte Modell unterteilt sich in hierarchische Ebenen, die flexibel erweitert werden können, um die Komplexität und die Feinheiten des Modells zu erhöhen. Ein exemplarisches Beispiel besteht aus drei Ebenen. Auf Ebene 1 befindet sich das Flugzeug, auf Ebene 2 die Kabine und das Antriebssystem und auf Ebene 3 die einzelnen Systeme des Antriebs, wie z.B. das Brennstoffzellensystem, die zusammen das gesamte Antriebssystem bilden. Dies ist eine der ersten Konfigurationen, die MIWa erstellt hat. Das Ziel ist es, Konfigurationen vergleichen zu können, z.B. der Ersatz des Kerosinantriebs durch einen Wasserstoffantrieb ersetzt wird.

Das Hauptziel der Abschlussarbeit besteht darin, durch gezielte Änderungen der Ausgangsparameter der Submodule und durch die Integration von Alternativmodellen. Dies wird durch die Anwendung einer ausgewählten Sequenz von Modelle erreicht, die in Abbildung 6 dargestellt ist.

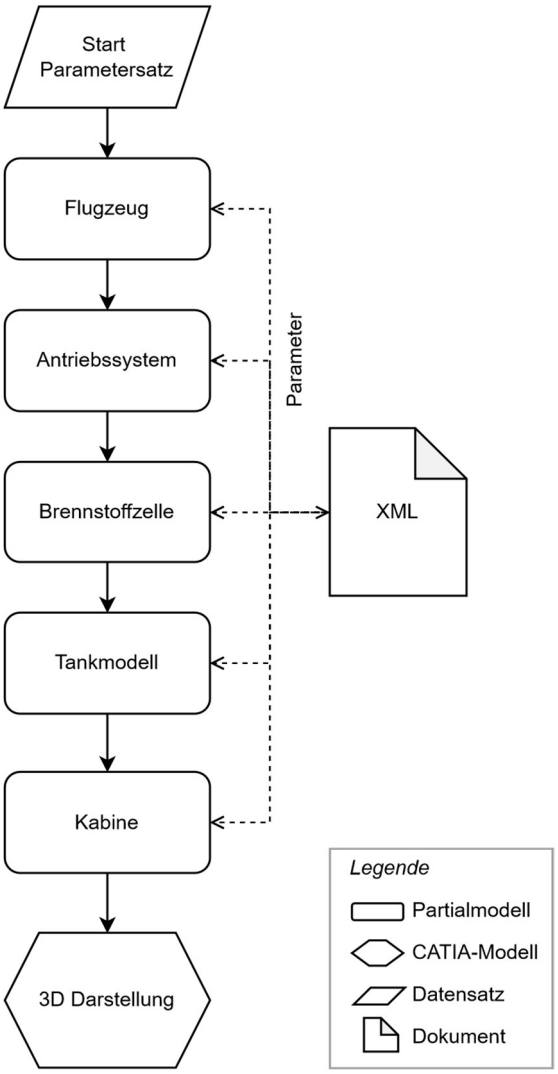


Abbildung 6: Reihenfolge der Modelle zu Erzeugung eines Simulationsvorgang (von M. Fuchs, 2023, DLR).

Die Beachtung dieser Reihenfolge ist von entscheidender Bedeutung, da jedes Modell in Abhängigkeit zu den Ergebnissen bzw. Parametern der vorangegangenen Modelle steht. Zur Veranschaulichung dieser Abhängigkeiten wird auf Abbildung 7 verwiesen, aus der sich ableiten lässt, dass die Abmessungen und die Anordnung des Tanks einen direkten Einfluss auf die Gestaltung und Größe der Kabine haben.

Zur Generierung eines Parametersatzes wird zunächst ein Anfangsparametersatz benötigt. Anschließend wird ein übergeordnetes Modell gestartet, das die wesentlichen Werte für die Berechnung des gewählten Antriebssystems (Kerosin-, Wasserstoff- oder Elektroantrieb) liefert. Anschließend kann das zugehörige Energieversorgungssystem ausgewählt werden, z.B. ein Hilfstriebwerk oder ein Brennstoffzellensystem. Daraus wird die benötigte Wasserstoffmenge berechnet, die auf abgeschätzten Leistungsanforderungen basiert, die wiederum für die Auslegung eines oder mehrerer Tanks relevant ist. Nach der Vordimensionierung des Tanks eine geeignete Kabine unter Berücksichtigung der gewünschten Passagierzahl und des zur Verfügung stehenden Bauraums entworfen werden. Die simulierte Variante ermöglicht schließlich die Visualisierung der Konfiguration in einer räumlichen Darstellung. Es ist wichtig zu betonen, dass die Verbindung zwischen den einzelnen

Submodellen durch die Verwendung von XML-Dateien (siehe Kapitel 2.2) realisiert wird. Jedes Teilsystem extrahiert seine Ausgangsparameter aus dieser Datei und schreibt die resultierenden Werte in dieselbe Datei zurück.

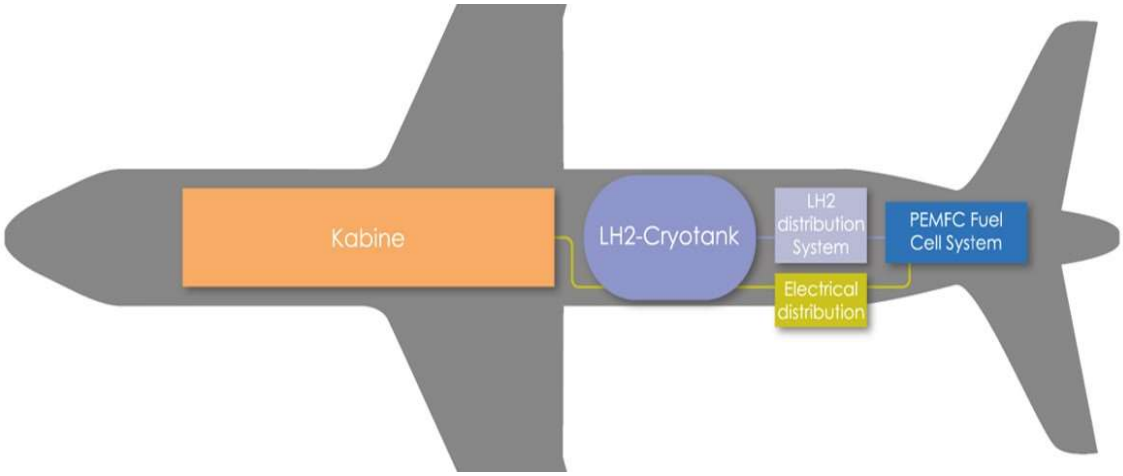


Abbildung 7: Schematische Darstellung der ersten Konfiguration (von M. Fuchs, 2023, DLR).

In der aktuellen Projektphase erfolgt die manuelle Erstellung der Startparameter in der entsprechenden XML-Datei. Die ausgewählten Submodelle werden im MCSE geöffnet und die XML-Datei wird in das Verzeichnis des jeweiligen Modells kopiert. Anschließend wird die Simulation gestartet und dieser Vorgang so lange wiederholt, bis die gewünschte Abfolge der Modelle simuliert wurde. Dies macht die Aufgabe zeitaufwendig und bietet ein großes Potenzial für Automatisierung, wodurch die Möglichkeit geschaffen wird, Simulationen zu wiederholen, um eine optimale Konfiguration von Komponenten zu erhalten, die die Anforderungen des Modells erfüllen.

Die genannten Teilmodelle sind in Abbildung 7 dargestellt und werden im Folgenden näher erläutert.

3.1 Brennstoffzellen Modell

Eines der initialen Modelle, das im ersten vereinfachten manuellen Durchlauf des Gesamtmodells und der ersten Variation entworfen wurde, beinhaltete die Substitution des Hilfstriebwerk (APU) durch ein Brennstoffzellensystem. In diesem Szenario basiert der Hauptantrieb jedoch weiterhin auf Kerosin. Die APU ist ein kleines Hilfstriebwerk, der unabhängig vom Hauptantriebssystem arbeitet und dazu dient, elektrische Energie, hydraulische Energie oder Druckluft für verschiedene Funktionen im Flugzeug bereitzustellen, wenn der Hauptantrieb nicht benutzt wird [6, S. 805–824].

Die Modellierung des Ersatzmodells basiert auf Polymer-Elektrolyt-Membran-Brennstoffzellen. Die Entwickler dieser Art von Brennstoffzellen, Nafion (@Dupont), sind die bekannteste und etablierteste Hersteller. Die wurden seit den 1960er Jahren in mehreren Varianten entwickelt und ist immer noch der Maßstab, an dem andere gemessen werden, und ist in gewisser Weise ein „Industriestandard“. Die verschiedenen Unternehmen, die Polymerelektrolytmembranen herstellen, haben ihre eigenen speziellen Tricks, von denen die

meisten urheberrechtlich geschützt sind. Ein gemeinsames Thema ist jedoch die Verwendung von sulfonierten Fluorpolymeren, normalerweise Fluorethylen [17].

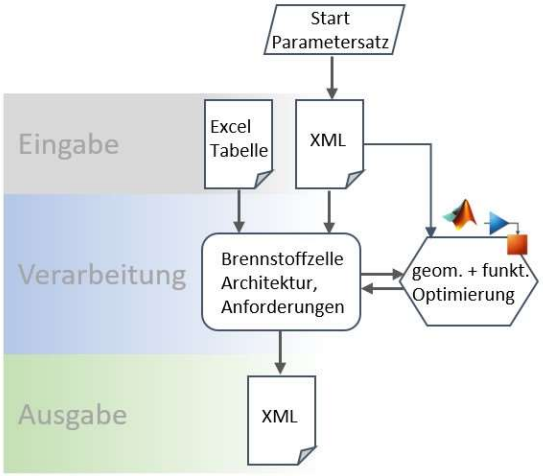


Abbildung 8: Modellstruktur für die Brennstoffzellenauslegung (von M. Fuchs, 2023, DLR).

Wie bereits erläutert, basieren alle Modelle auf dem EVA-Prinzip (siehe Kapitel 2.1) und sind in die drei Phasen *Eingabe, Verarbeitung und Ausgabe* unterteilt, wie in Abbildung 8 dargestellt. Zur Berechnung, Optimierung und Dimensionierung der Brennstoffzelle wird eine Schnittstelle von Magic Cyber System Engineer (siehe Kapitel 2.1) zu MATLAB verwendet. Diese Schnittstelle ist eine Erweiterung des Modellierungsprogramms. *„MATLAB ist eine Programmierplattform, die speziell für Ingenieure und Wissenschaftler entwickelt wurde, um Systeme und Produkte zu analysieren und zu entwerfen, die unsere Welt verändern. Das Herzstück von MATLAB ist die MATLAB-Sprache, eine matrixbasierte Sprache, die den natürlichsten Ausdruck von Computermathematik ermöglicht.“* [39].
Sämtliche Eingabewerte werden durch den Nutzer als Parametersatz in der XML-Datei abgelegt. Zu den Anfangswerten des Modells gehören unter anderem die Systemleistung und die Missionsbeschreibung. Nach Durchlaufen des Modells werden optimale Werte für die Anzahl der Brennstoffzellen, das benötigte Wasserstoffvolumen und die Bauteilabmessungen ermittelt und in der XML-Datei gespeichert.

3.2 Tank Modell

Bei der Entscheidung über die Integration von Wasserstoff in bestimmte Flugzeugkomponenten müssen das Volumen und der Bauraum des Wasserstofftanks berücksichtigt werden. Dies stellt die Hauptaufgabe dar, welsche durch das Tankmodell erfüllt wird. In dieser Phase der Simulation wird eine vorläufige Dimensionierung der Wasserstoffspeicheranlage erstellt. In dieser Auslegung werden erhebliche Vereinfachungen vorgenommen. Beispielsweise werden Komponenten wie Ventile nicht berücksichtigt. Der Schwerpunkt der Simulation liegt auf der Generierung grober Parameter eines Wasserstoffspeichers aus einem Zielvolumen. Das Zielvolumen wird als Eingabe aus der vorherigen Simulation ausgelesen und in ein Volumen flüssigen Wasserstoffs umgerechnet.

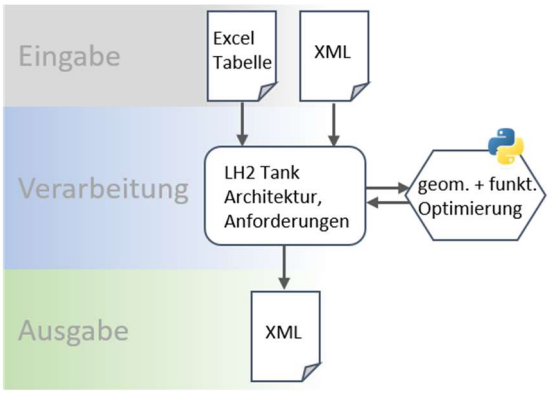


Abbildung 9: Modellstruktur für die Tankauslegung (von M. Fuchs, 2023, DLR).

Nach der Extraktion der Anfangsparameter, einschließlich der Materialeigenschaften des gewählten Werkstoffes, wird eine Optimierung gestartet, um die minimale Oberfläche des Tanks zu finden, wie in Abbildung 9 dargestellt. Diese Optimierung wird durch ein Python-Skript ausgeführt, das vom MCSE aufgerufen wird. Die Ergebnisse dieses Submodells führen zu einem Tank mit den beschriebenen Parametern bspw. Länge, Durchmesser, Wanddicke und Position.

3.3 Kabinen Modell

Das bisher letzte Teilmodell modelliert, die gesamte Kabine anhand von bereits in der XML-Datei vordefinierten Daten, wie z.B. die gewünschte Anzahl an Passagiere, die Länge des Tanks und die verfügbare Leistung für die einzelnen Komponenten der Kabine. Das Modell besteht aus einem deskriptiven und einem analytischen Modell. Der erste Teil wird auf Basis von SysML (siehe Kapitel 2.1) und in MCSE aufgesetzt. Auf diese Weise kann Magic Cyber Systems Engineer die Anforderungen des Modells überprüfen und bestätigen.

Das zweite Teilsystem, der analytische Teil, konzentriert sich auf die geometrische Auslegung der Kabine in Matlab. Dabei werden Komponenten wie z.B. Sitze in einem dreidimensionalen Raum positioniert, wie in Abbildung 10 dargestellt.

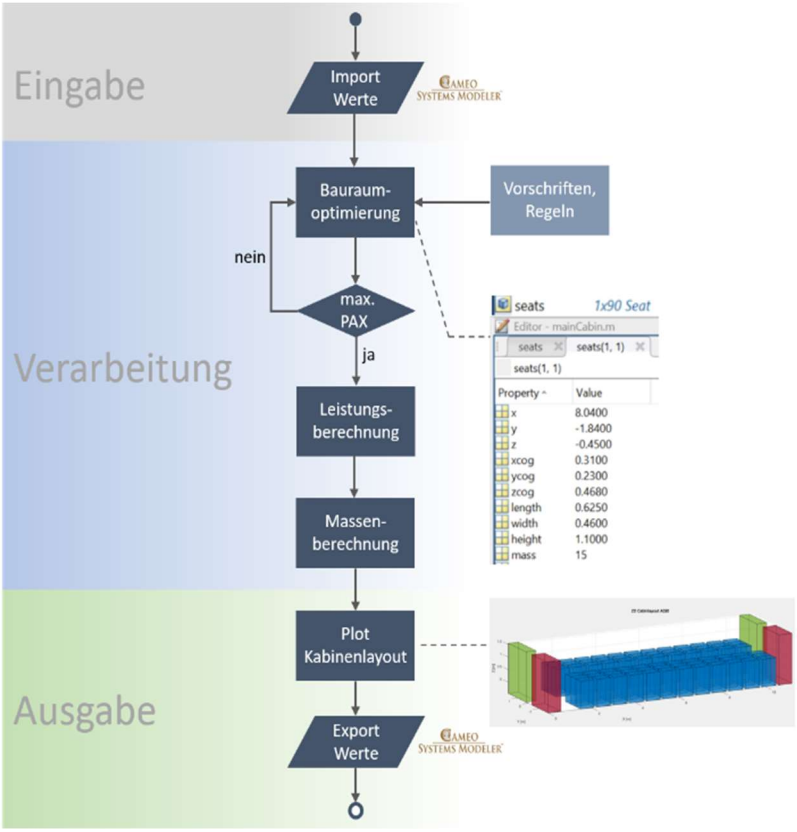


Abbildung 10: Modellstruktur für die Kabinenauslegung (von M. Fuchs,2023, DLR).

Der Entwurf wird in einer MATLAB-Umgebung gestartet, wobei die Anfangsparameter wie Tanklänge und Leistung bereits im Eingabedatensatz festgelegt sind. Hieraus resultieren neben der Anzahl an Komponenten der Kabine mit der optimalen Passagieranzahl auch der Massenschwerpunkt und eine überschlägige Leistungsberechnung.

3.4 Weitere Modelle

Derzeit wird an verschiedenen Modellen gearbeitet, die zu realitätsnäheren Flugzeugvarianten führen sollen Ein Beispiel dafür ist das Modell eines mit Wasserstoff statt mit Kerosin betriebenen Triebwerks. Andere Modelle, wie zum Beispiel ein übergeordnetes Modell, zielen darauf ab, Missionsparameter eine Flugmission, wie zum Beispiel der Takeoff-Time oder der benötigte Schub in verschiedenen Phasen zu berechnen. Ein weiteres Modell, das sich noch in der Entwicklung befindet, beschäftigt sich mit der Erstellung von Rohrleitungen für die automatische Verbindung von Systemen, z.B. zwischen dem Wasserstofftank und dem Brennstoffzellensystem.

Die Auswahl verschiedener Modellvarianten zur Generierung verschiedener Flugzeugvarianten kann mühsam sein, obwohl dies halbautomatisch erfolgen kann. Durch die Festlegung der Reihenfolge der zu simulierende Modelle wird die Erstellung der Anfangsparameter in einer XML-Datei einfacher und effizienter. Dies eröffnet die Möglichkeit, eine grafische Benutzeroberfläche zu entwickeln, um den Prozess der Modellsimulation zu beschleunigen und

die Erstellung der Datei mit den Startparametern zu vereinfachen. Dies trägt wesentlich zur Reduzierung von Fehlern bei, die bei der manuellen Erstellung einer XML-Datei auftreten können. Die grundlegende Architektur der Anwendung zu klären, damit alle identifizierten Anforderungen erfüllt werden können.

4 Definition der vereinfachten Basis-Architektur

In der Basisarchitektur werden Entscheidungen über Architekturkonzepte und konkrete Technologien festgelegt. Diese dienen als Rahmen für das Projekt und bestimmen den Projektumfang. Die Definition der Architektur ist sehr wichtig, da die Anforderungen von der technischen Lösung getrennt werden. Die Architektur wird im Rahmen der Bachelorarbeit durch Skizzen, Blockdiagramme und Erläuterungen definiert [35, S. 51–53].

4.1 Erkennung des Stakeholders mit Bedürfnissen

Es ist von grundlegender Bedeutung, dass die Stakeholder das geplante Programm akzeptieren und bis zu einem gewissen Grad in das Projekt involviert sind. Die Erfüllung der Bedürfnisse der Mehrheit der Stakeholder ist für den Erfolg des Projekts entscheidend. Ein Stakeholder vertritt die Interessen einer Organisation oder eines Projekts und kann direkt oder indirekt von der Anwendung betroffen sein. Ein Stakeholder ist eine Person, die von einer Entscheidung oder Maßnahme des Systems betroffen ist. Die Definition der Stakeholder sollte unbedingt in einem Workshop erfolgen, in dem alle am Projekt beteiligten Partner ihre Perspektive und ihr Fachwissen einbringen können [35, S. 54–58]. Als Ergebnis des Projektworkshops werden verschiedene Personas identifiziert und deren spezifische Bedürfnisse dokumentiert. Eine detaillierte Übersicht ist in Abbildung 11 dargestellt. Die Bedürfnisse wurden in Form von Personas und User Stories aufgelistet und im Folgenden beschrieben:

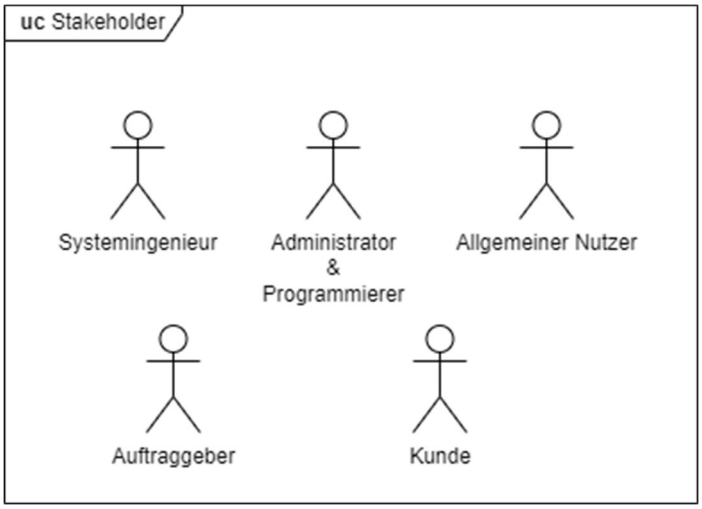


Abbildung 11: Vereinfachtes Use Case Diagramm mit Stakeholder.

4.1.1 Systemingenieur

Der Systemingenieur ist eine Schlüsselperson in dieser Anwendung. Dieser ist verantwortlich für die Erstellung der MCSE-Modelle und deren Verknüpfung mit anderen Modellen. Ein Systemingenieur ist ein Fachexperte, der die SysML in diesem Kontext einsetzt, um komplexe

Systeme zu analysieren, zu entwerfen und zu simulieren. Es folgt eine Liste der formulierten Users-Stories:

1. Der Systemingenieur soll mit der GUI in der Lage sein, nacheinander mehrere Magic Cyber Systems Engineer Modelle starten und dessen interne Simulationskonfigurationen automatisch ausführen können.
2. Der Systemingenieur möchte in der GUI die Möglichkeit haben auszuwählen, welche (Magic Cyber Systems Engineer) Modelle gestartet und in welcher Reihenfolge diese ausgeführt werden sollen.
3. Der Systemingenieur möchte das die GUI zum Start eine leere Universal-XML aus einer Vorlage erzeugt und verwendet.
    - Er möchte das die GUI diese als erstes mit Daten wie Datum, Versions Nummer und Name ausfüllt.
    - Er möchte, dass die GUI nach jedem erfolgreichen, durchgeführten Start eines (Magic Cyber Systems Engineer) Modells den Namen des Modells in den Header der XML schreibt, damit nachvollzogen werden kann, welche Modelle alle für die Version genutzt werden.
    - Der Systemingenieur wünscht sich, dass am Ende des Durchlaufs die finale XML in einem Verzeichnis abgespeichert wird.
4. Der Systemingenieur würde sich wünschen, wenn die GUI ihm am Ende mitteilt, dass die Versionsauslegung erfolgreich war oder nicht.
5. Der Systemingenieur möchte mindestens die Position und Anzahl der Tanks bestimmen können, um direkt Einfluss auf die Verteilung der Treibstoffmasse und benötigten Volumens innerhalb des Flugzeugs nehmen zu können.
6. Der Systemingenieur wünscht sich die Möglichkeit eine Vergleichssimulation aufsetzen zu können, in der er mindestens zwei Varianten konfigurieren kann, um nach der Simulation einen detaillierten Vergleich als Ergebnis zu erhalten.
7. Der Systemingenieur möchte die zu simulierenden Variantenoptionen aus Drop-Down-Feldern auswählen können, um verschiedene Auswahlmöglichkeiten nur bei Bedarf angezeigt zu bekommen und die GUI übersichtlich zu halten
8. Der Systemingenieur möchte Daten seiner Wunschmission in Freitextfelder eingeben können, um diese für die Auslegung genau und flexibel definieren zu können.
9. Der Systemingenieur wünscht, dass die Anbindung von Startparametern und externen Datenbanken wie z.B. Excel-Tabellen oder CSV-Dateien möglich ist.

4.1.2 Allgemeiner Nutzer

Der allgemeine Benutzer wird als eine Person beschrieben, die möglicherweise nicht über ausreichende Fachkenntnisse über die einzelnen Teilsysteme verfügt. Dennoch sollte Stakeholder in der Lage sein, korrekte und verständliche Variationen zu erzeugen. Nachfolgend sind die formulierten Use Stories aufgelistet:

10. Der Nutzer wünscht sich eine Rückmeldung der GUI wenn Fehler auftreten.
11. Der Nutzer möchte eine Auswahl über die genutzten Modellparameter treffen können, mit Unterscheidung in Standard-Anwendung und erweiterte Einstellmöglichkeiten.
12. Der Nutzer möchte, dass nach Eingabe der Missionsdaten und der Variantenkonfiguration die Datenspeicherung in der XML und die Ansteuerung der notwendigen Modelle bis hin zur Ergebnisausgabe und 3D-Visualisierung automatisch

abläuft, um einen nutzerfreundlichen und professionellen Programmdurchlauf zu erhalten.
13. Der Nutzer wünscht sich eine übersichtliche und intuitive GUI-Oberfläche, die schnell verständlich ist, um eine einfache Bedienung zu gewährleisten.
14. Der Nutzer wünscht sich eine Visualisierung der getroffenen Auswahl, um diese sinnvoll überblicken zu können.

4.1.3 Administrator

In diesem Rahmen der Abschussarbeit ist der Administrator auch der Anwendungsentwickler und für die Wartung und Erweiterung der Anwendung verantwortlich. Eine Liste der User Stories des Verwalters findet sich nachfolgend:

15. Der Administrator wünscht sich die Möglichkeit, dass Personen ohne Programmierkenntnisse neue Modelle und Submodule zur Anwendung hinzufügen können, ohne dabei auf den Quellcode der Anwendung zugreifen zu müssen.
16. Der Administrator wünscht sich eine einfache Anwendungsstruktur (Ordner Struktur) für die Wartung und Erweiterung der Anwendung, in der auch der Quellcode gut strukturiert ist.
17. Der Administrator möchte, dass die Integration generisch entwickelt wird, um sicherzustellen, dass die Anwendung auch bei einem Wechsel des Hauptnutzers oder des Hauptmodells ihren Zweck erfüllt.
18. Der Administrator wünscht sich, dass alle von der Anwendung generierten Meldungen in einer externen Datei gespeichert werden, um Fehler nachverfolgen zu können.

4.1.4 Weitere Stakeholder

Aufgrund der bereits definierten Aufgabenstellung, der zeitlichen Begrenzung und der umfangreichen Inhalte war es nicht möglich, weitere Personas, wie z.B. potenzielle Kunden, zu beschreiben und zu dokumentieren. Die Anwendung soll jedoch erweiterbar sein, um zukünftige Anforderungen und Bedürfnisse berücksichtigen zu können.

4.2 Aufbau und Priorisierung der Anforderung

Eine Anforderung bildet die Grundlage für die Systementwicklung durch die Festlegung, welche Leistungen das System erbringen soll. Diese Anforderungen werden aus den individuellen Bedürfnissen der Stakeholder abgeleitet [35, S. 59–61].

Nach der Zusammenstellung der Bedürfnisse wurden diese in

Tabelle 1 übertragen und für die anschließende Bewertung vorbereitet. Die Bewertung spielt eine entscheidende Rolle bei der Priorisierung und Bearbeitung der resultierenden Anforderungen. Für die Bewertung wird eine unkomplizierte Methode mit fünf Kategorien angewendet, wobei jeder Kategorie eine Punktzahl von 0 bis 10 zugeordnet wird. Alle Kategorien werden in diesem Fall gleich gewichtet, basierend auf der ursprünglichen Erfahrung des Programmierers. Nach der Bewertung jeder Kategorie wird die Gesamtpunktzahl ermittelt, aus der die sich ergebenden Anforderungen abgeleitet werden. Die Kategorien werden in einer einzigen Runde bewertet und sind wie folgt definiert:

- Der Benutzerwert ist die Funktionalität, die der Anwender gewinnen kann.
- Die Umsetzbarkeit ist der Schwierigkeitsgrad der Aufgabe für den Programmierer.
- Die Risiken ist die Unzufriedenheit der Nutzer, wenn die Anforderungen nicht erfüllt werden.
- Die Abhängigkeiten sind Beziehungen zu anderen Anforderungen
- Langfristige Auswirkungen beschreibt den negativen Einfluss, wenn die Anforderung nicht erfüllt wird.

Tabelle 1: Liste der Anforderung des Projektes mit Auswertung.

	Anforderung	**Benutzerwert**	**Umsetzbarkeit**	**Risiken**	**Abhängigkeiten**	**Langfristige Auswirkungen**	**Summe**
1	Automatisch MCSE-Modelle mit den entsprechenden Konfigurationen ausführen zu können.	10	10	10	10	10	50
2	Auswahl der auszuführenden Modelle.	10	10	10	10	10	50
3	Generierung einer XML-Datei für den Start des Simulationslaufs	10	10	10	10	10	50
3.1	Einführung eines Zeitstempels und einer Versionsnummer in der XML-Anfangsdatei	9	10	8	8	10	45
3.2	Einführung der Namen der Simulationsmodelle in der XML-Datei	10	10	8	8	9	45
3.2	Verschieben der XML-Datei in das gewünschte Verzeichnis.	8	10	3	5	8	34
4	Kommunikation der Prozesse über die GUI	10	10	10	10	10	50
5	Positionierung der Komponenten über die GUI	5	10	0	2	3	20
6	Auswahl einer beliebigen Anzahl von Simulationen	10	10	10	10	10	50
7	Parametervariationen ermöglichen	8	10	10	8	10	46
8	Freitext-Eingabemöglichkeiten für Eingabeparameter anbieten	10	10	10	10	10	50
9	Anbindung externer Datenbanken für die Startparameter der Modelle	8	10	8	4	10	40
10	Mitteilung von Fehlermeldungen über die GUI	10	10	10	10	10	50
11	Auswahl zwischen Standard- und Expertenmodus	10	10	10	10	10	50
12.1	Einfügen eines Play-Buttons zum Starten der Simulationen	8	10	10	7	5	40
12.2	3D-Visualisierung der Varianten	10	4	7	8	10	39
13	Intuitive und benutzerfreundliche Gestaltung	10	10	10	10	10	50
14	Visualisierung der ausgewählten Modelle	10	10	10	10	10	50
15	Einfache Erweiterung und Pflege von Modellen und Funktionalitäten	10	10	10	10	10	50
16	Reproduzierbare und einfache Ordnerstruktur zur Erweiterung der Modelle	10	10	10	10	10	50
17	Möglichst neutrale und erweiterbare Programmierung zur allgemeinen Nutzung und Wiederverwendung der Anwendung	10	8	9	10	10	47
18	Erstellung einer Log-Datei für alle generierten Meldungen	5	10	9	8	10	42

Anschließend werden die Anforderungen nach ihrer Bewertung geordnet und, wie in Tabelle 2 dargestellt, neu formuliert, wobei alle Anforderungen, die eine Summe von mehr als 45 ergeben, mit einer Priorisierung angenommen wurden.

Tabelle 2: Anforderungstabelle mit Priorisierung.

	Anforderung	**Kategorie**
1	Automatisch MCSE-Modelle mit den entsprechenden Konfigurationen ausführen	must
5	Auswahl einer beliebigen Anzahl von Simulationen	must
3	Generierung einer XML-Datei für den Start des Simulationslaufs	must
7	Mitteilung von Fehlermeldungen über die GUI	must
15	Einführung eines Zeitstempels und einer Versionsnummer in der XML-Anfangsdatei	must
8	Auswahl zwischen Standard- und Expertenbenutzer haben	must
9	Intuitive und benutzerfreundliche Gestaltung	must
10	Visualisierung der ausgewählten Modelle	must
11	Einfache Erweiterung und Pflege von Modellen und Funktionalitäten	must
12	Reproduzierbare und einfache Ordnerstruktur zur Erweiterung der Modelle	must
13	möglichst neutrale und erweiterbare Programmierung zur allgemeinen Nutzung und Wiederverwendung der Anwendung	must
14	Parametervariationen zur Verfügung stellen	must
16	Einführung der Namen der Simulationsmodelle in der ursprünglichen XML-Datei	should
17	Erstellung einer Log-Datei für alle generierten Meldungen	should
6	Freitext-Eingabemöglichkeiten für Eingabeparameter anbieten	should
2	Auswahl der auszuführenden Modelle zu haben.	should
4	Kommunikation der Prozesse über die GUI	should
18	Anbindung externer Datenbanken für die Startparameter der Modelle	should
19	Einfügen eines Play-Buttons zum Starten der Simulationen	should
20	3D-Visualisierung der Varianten	should
21	Verschieben der XML-Datei in das gewünschte Verzeichnis.	should
22	Positionierung der Komponenten über die GUI	should

SysML bietet die Möglichkeit, Anforderungen grafisch darzustellen. Diese ist in Abbildung 12 gezeigt. Das abgebildete Diagramm wird als Anforderungsdiagramm bezeichnet. Mithilfe dieses Diagramms ist es möglich, Beziehungen zwischen Anforderungen herzustellen und Anforderungen zu kategorisieren [35, S. 314–315]. In dieser Arbeit wird jedoch vereinfacht, da die Komplexität des Systems nach Ansicht des Entwicklers diesen Detaillierungsgrad nicht erfordert. Die visuelle Darstellung der ersten Anforderung erwies sich als entscheidend für den Programmierprozess, insbesondere da Anforderung Nr. 1 in

Tabelle 2 die Benutzbarkeit der Anwendung maßgeblich beeinflusst und somit als zentrale Herausforderung des Projekts gilt. Das Anforderungsdiagramm ermöglicht die Spezifizierung weiterer erforderlicher Informationen, z.B. der zulässigen Schritte, um ein MCSE-Modell richtig auszuführen.

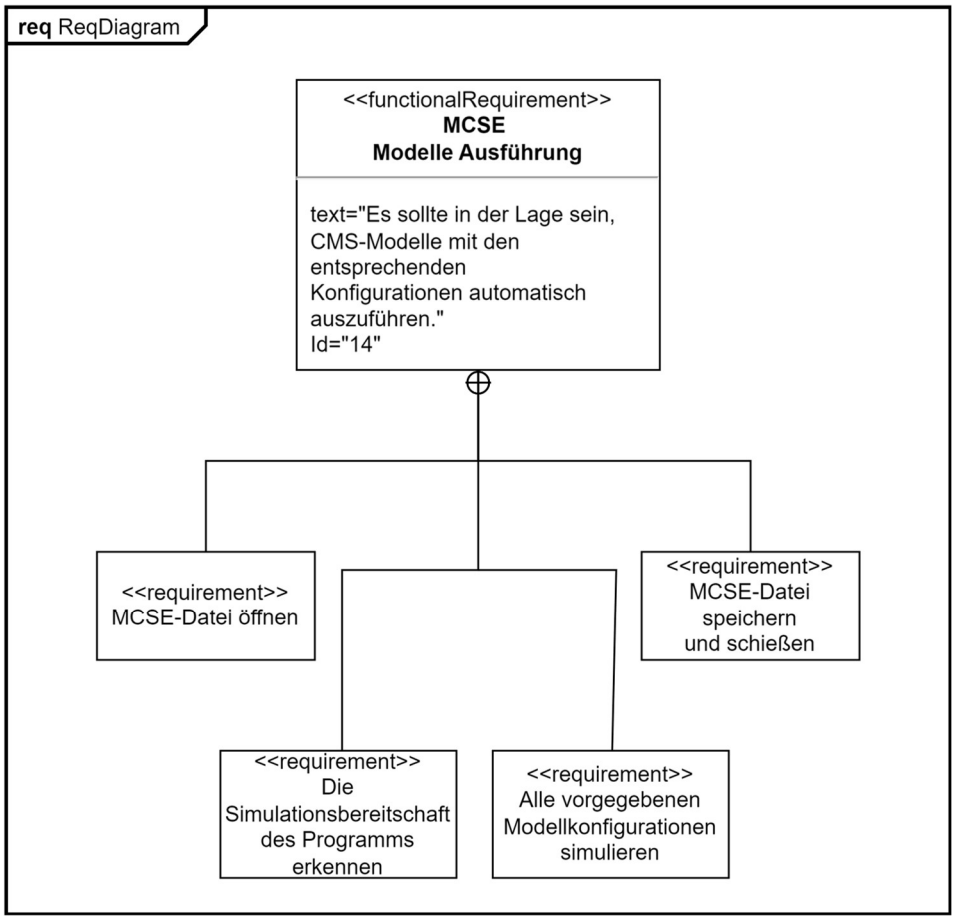


Abbildung 12: Vereinfachtes Anforderungsdiagramm.

4.3 Erstellung der Anwendungsfälle

Nach der Definition der Anforderungen können diese durch die Erstellung von Anwendungsfällen des Systems erfüllt werden. Ein Anwendungsfall, auch Use Case genannt, beschreibt eine Reihe von Aktionen eines Systems, die zu einem bestimmten Ergebnis führen. Dieses Ergebnis hat in der Regel einen bestimmten Wert für die Stakeholder. SysML und UML bieten durch das Use Cases Diagramm eine Übersicht der beteiligten Benutzer und Use Cases. Ein solches Diagramm ist in Abbildung 13 dargestellt [35, S. 234 - 238].

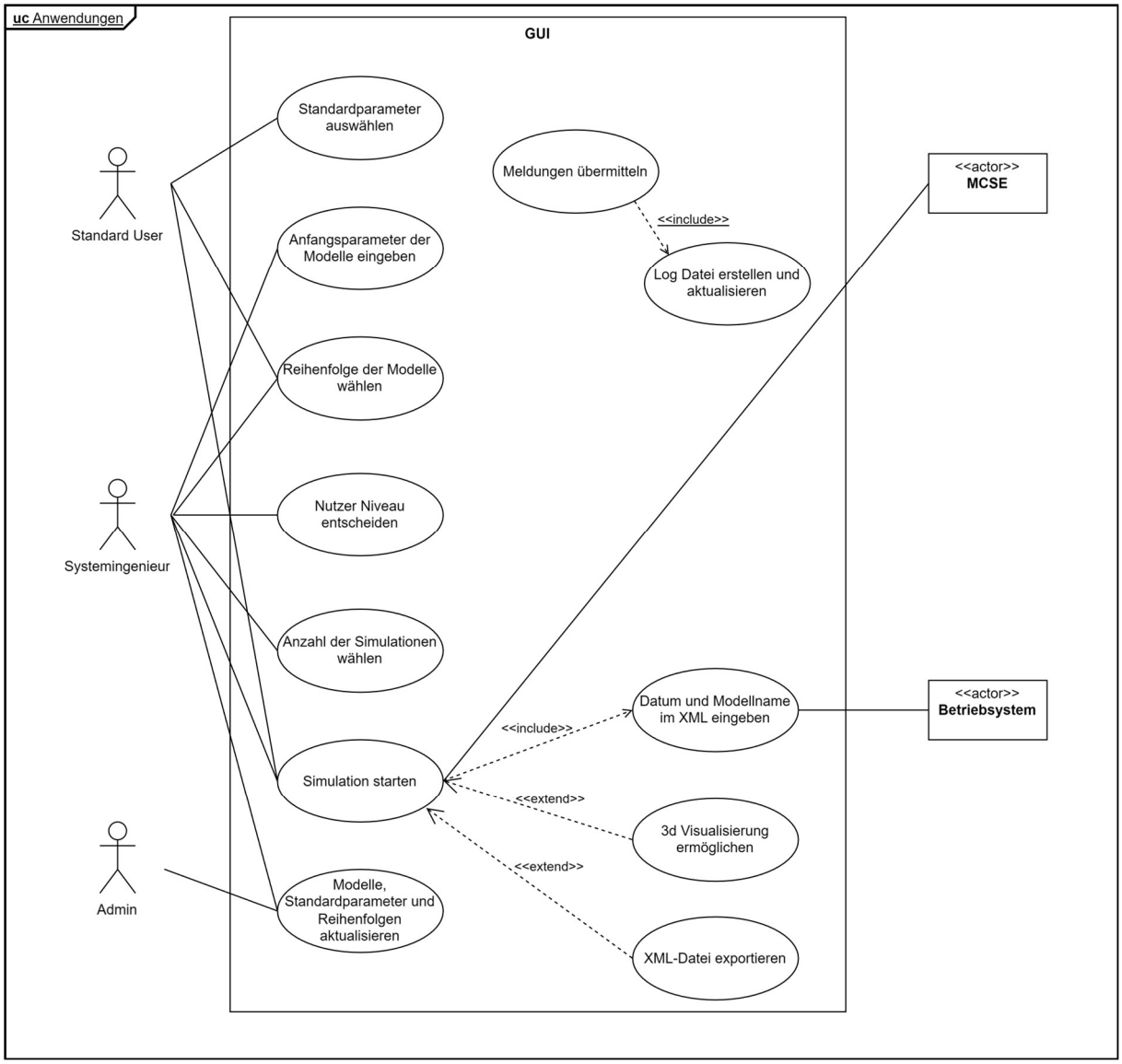


Abbildung 13: Use Case Diagramm der GUI.

Wenn die Design- und Programmieranforderungen vernachlässigt werden, können nahezu alle definierten Anforderungen durch die definierten Anwendungsfälle des Systems erfüllt werden. Außerdem ist es eine Verständnishilfe für Programmierer, um alle Anwendungsfälle zu implementieren, ohne den Überblick zu verlieren.

Zusammenfassend lässt sich sagen, dass die Anwendung eine Verbindung zwischen den Benutzern und Magic Cyber Systems Engineer herstellt, indem es Systemingenieuren und Administratoren ermöglicht, Modelle, Modellreihen und Parameter zu speichern. Auf diese Weise können andere Benutzer Simulationen durchführen und Systemauslegungen erstellen.

Zu beachten ist, dass der Anwendungsfall „Meldungen übermitteln“, der sich oben rechts in der Abbildung 13 befindet, bewusst keine direkte Verbindung zu den anderen Artefakten im Diagramm hat. Dies wurde strategisch gewählt, um das Diagramm übersichtlich zu halten, obwohl dieser Anwendungsfall in der Realität mit allen anderen Anwendungsfällen des Systems in Verbindung steht, da das System eine systematische Kommunikation ermöglichen soll.

Relevante Eigenschaften des Diagramms sind die „include"- und „extend“-Verbindungen. Ein Beispiel hierfür ist die „include“-Verbindung von „Meldungen übertragen“ zu „Logdatei erstellen und aktualisieren“, die signalisiert, dass beide Anwendungsfälle ausgeführt werden müssen, um einen reibungslosen Ablauf zu gewährleisten. Im Gegensatz dazu hat der Anwendungsfall „Simulation starten“ zwei „extend“-Verbindungen, die auf bestimmte Bedingungen hinweisen, die für die entsprechende Funktionalität erfüllt sein müssen. In diesem Szenario kann der Benutzer am Ende der Simulation wählen, ob die XML-Datei exportiert werden soll oder nicht.

5 Systemaufbau

Nach der Festlegung der grundlegenden Architektur erfolgt die Darstellung der konzeptionellen Lösung mit Hilfe von UML-Diagrammen, insbesondere des Klassendiagramms. Das Klassendiagramm ist eines der am häufigsten verwendete UML-Diagramme und bildet die Grundlage für objektorientierte Lösungen. Es visualisiert die Klassen im System, ihre Attribute, ihre Beziehungen zueinander, sowie die Verwendung zwischen den einzelnen Klassen. Die Gruppierung von Klassen ermöglicht die Erstellung von Klassendiagrammen, vorwiegend wenn umfangreiche Systeme in Diagrammform dargestellt werden sollen [20, S. 14-15].

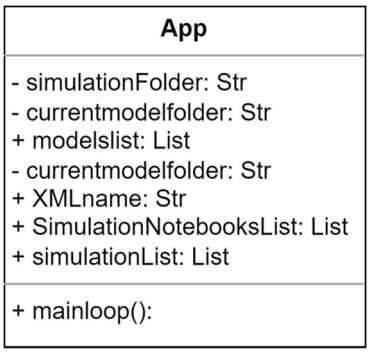


Abbildung 14: Darstellung der Klasse „App“ im Klassendiagramm.

Abbildung 14 zeigt eine Klasse in einem Klassendiagramm mit dem Namen „App“ im Heades und den Eigenschaften dieser Klasse, wie z.B. „simulationFolder“, darunter. Zur besseren Verständlichkeit wird auch die Art der Variable deklariert, in diesem Fall als String. Die Symbole vor den Eigenschaften geben Auskunft darüber, ob es sich um öffentliche (+) oder nicht öffentliche (-) Eigenschaften handelt. Dies legt fest, ob andere Klassen auf diese Eigenschaften zugreifen oder dieser verändern können. Schließlich werden die Funktionen bzw. in Python-Notation die Definitionen aufgelistet, die das Objekt „App“ besitzt. In Python werden die Begriffe Klasse und Objekt gleichbedeutend verwendet.

Nach Kenntnis des Grundkonzepts eines Objekts wird das System wie in Abbildung 15 dargestellt. Die Anwendung besteht aus zwölf verschiedenen Klassen, die spezifische Funktionen und die in den vorhergehenden Kapiteln definierten Anforderungen erfüllen Das Diagramm stellt eine Vereinfachung dar, nicht alle Variablen der Klasse sind explizit aufgeführt. Stattdessen wird ein "..." Symbol verwendet. Dies zeigt, dass es zusätzliche Variablen und Funktionen gibt, die nicht im Diagramm dargestellt werden.

Abbildung 15 liefert die entscheidende Information, dass die Anwendung eine Struktur mit einer Hauptklasse, der Klasse „App“, aufweißt. Diese Hauptklasse fungiert als zentrales Bindeglied zwischen allen anderen Klassen innerhalb der Anwendung. Die Klassen „TextBox“ und „InfoFrame“, obwohl sie prinzipiell einen festen Zusammenhang mit den anderen Unterklassen des Systems haben, wurden in Abbildung 15 aus didaktischen Gründen vereinfacht dargestellt.

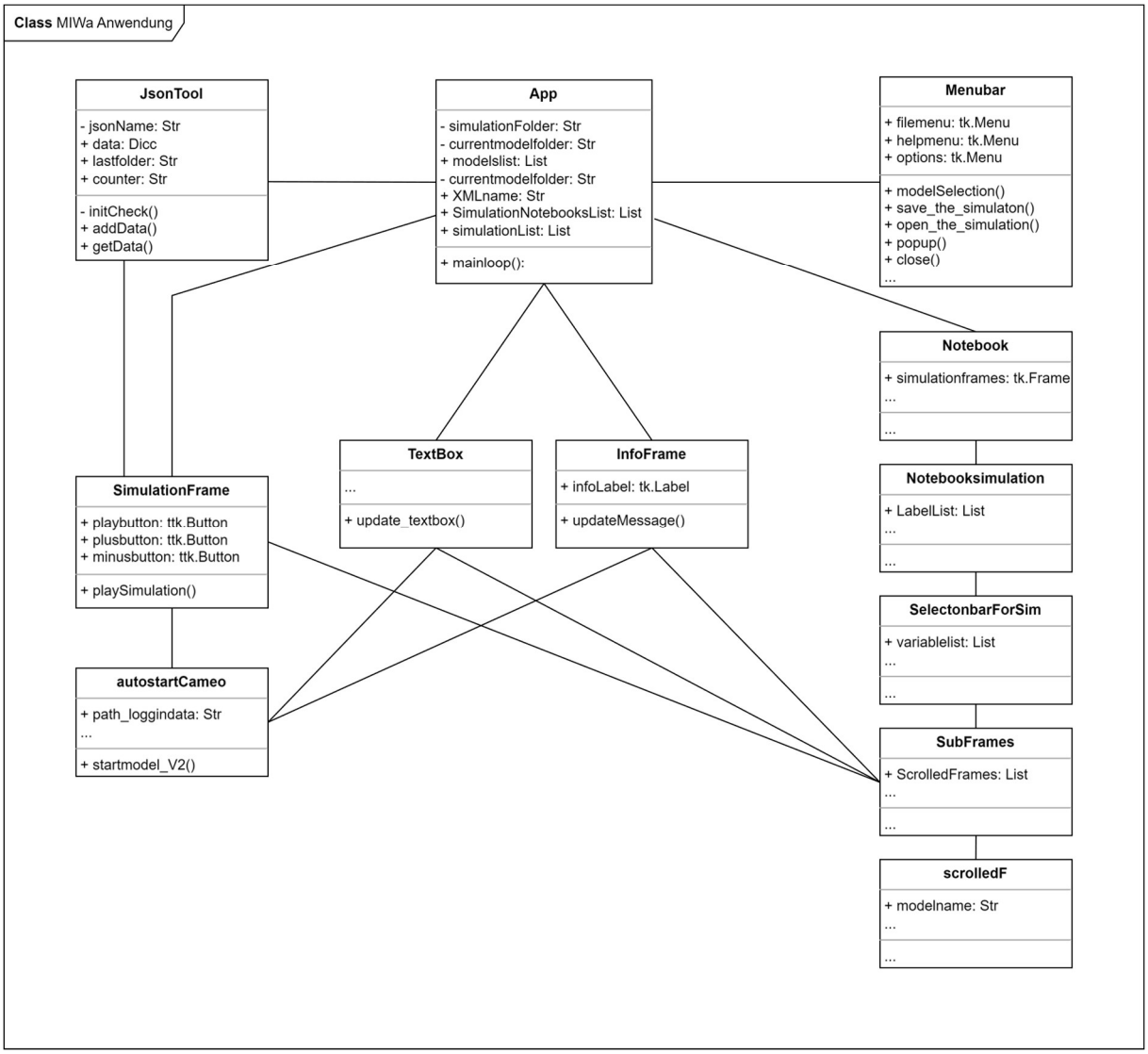


Abbildung 15: Vereinfachtes Klassendiagramm.

Zum besseren Verständnis der Anwendung dient Abbildung 16. Diese gibt einen Überblick über das Layout der Anwendung mit den entsprechenden Klassennamen. Die Aufgabe der meisten Kassen ist erkennbar. Eine ausführliche Erläuterung der zwölf Klassen und ihrer Funktionen folgt um besseres Verständnis der jeweiligen Struktur und der Erweiterungsmöglichkeiten in zukünftigen Aktualisierungen.

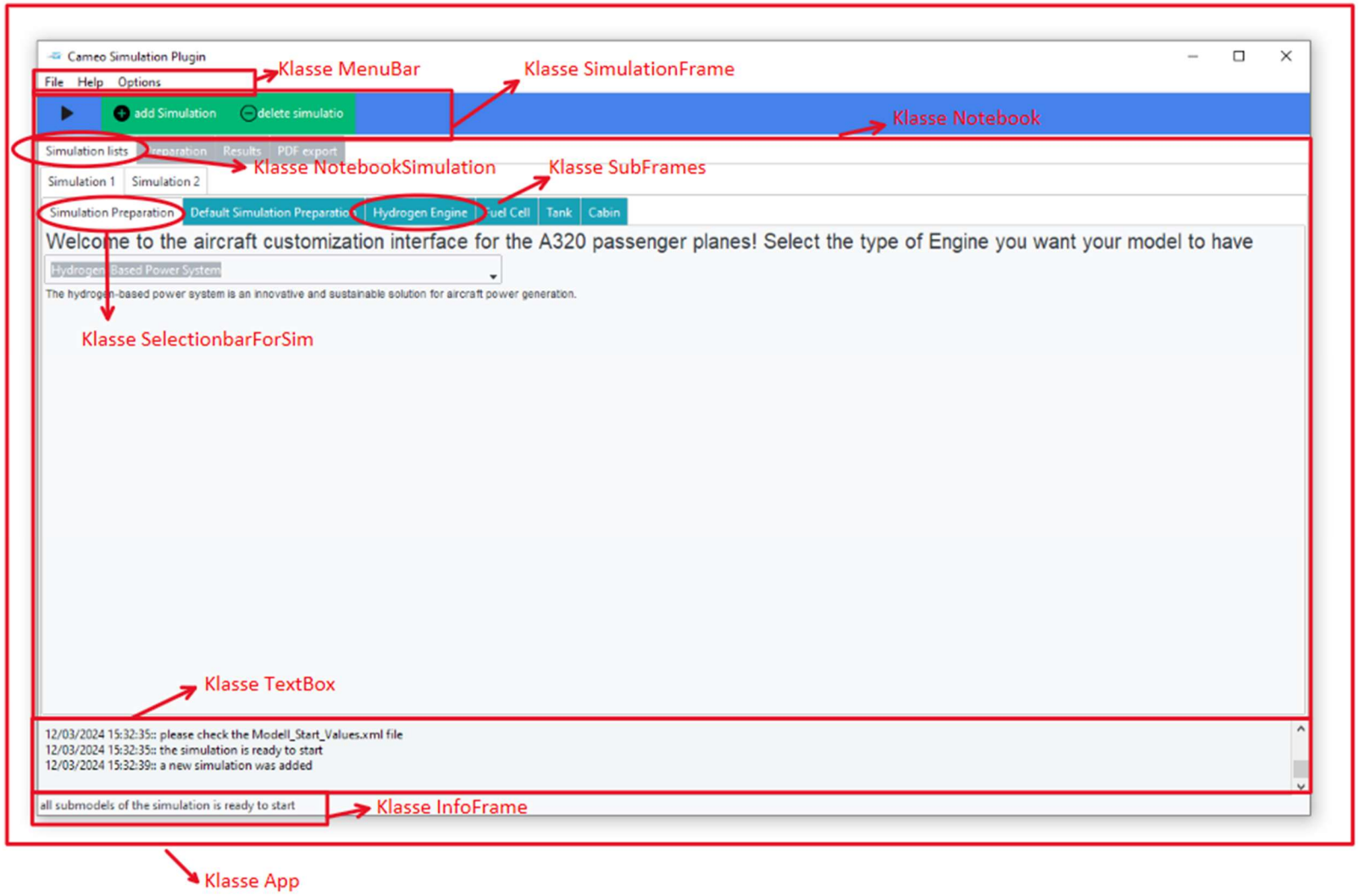


Abbildung 16: Erste Übersicht der Anwendung mit den zugehörigen Objekten

5.1 Klasse App

Wie bereits in Kapitel 2.3 erwähnt, hat die OOP die Eigenschaft Klassen zu vererben, was bei der Erstellung neuer Klassen von Vorteil ist. Diese Möglichkeit wird systematisch für fast alle Klassen der Anwendung genutzt, z.B. für die Klasse App, die die Eigenschaften der Klasse tkinter.tk übernimmt. Abbildung 17 zeigt einen Ausschnitt aus dem Code, in dem die Klassenvererbung implementiert ist.

```
50  class App(tk.Tk):
51  # this is the main class of the program
52  #
53  # the program contols the GUI, the logic of the program
54  # and had access to the data of the program
55      def __init__(self):
56          super().__init__()
57
58          """global variables"""
59          self.XMLexist = False
60          # hier will be the path of the folder selected by the user
61          self._simulationFolder = ""
62          self._currentmodelfolder = tk.StringVar()
63          self.modelslist = os.listdir(r"01_Models")
64          self._currentmodelfolder.set(f"01_Models\\Plane")
65          ######################
66
```

Abbildung 17: Klassenvererbung als Codeabschnitt.

In Zeile 50, innerhalb der Klammern, werden zusätzliche Informationen über die Klasse, in diesem Fall „tk.TK“, angezeigt. Diese Klasse stammt aus dem Modul "tkinter" und stellt ein Fenster für eine grafische Benutzeroberfläche (GUI) dar. Deshalb hat diese Kasse nur eine Definition (Abbildung 14), die vererbt wurde, die „mainloop“-Funktion. Diese ruft das Hauptfenster der Anwendung und wartet, bis der Benutzer damit interagiert. Diese Klasse soll nicht nur die Standardeigenschaften eines Fensters besitzen, sondern auch die Ordnerstruktur der Anwendung verwalten. Daher werden weitere Eigenschaften wie „_simulationFolder“ und andere entsprechend deklariert. Für den einfachen Zugriff auf Eigenschaften, die von anderen Klassen erzeugt werden, z.B. die Werte einer Simulation, wird eine Liste deklariert, die die verschiedenen Simulationen mit den Werten der Simulation enthält. Diese Liste heißt „SimulationNotebooksList“ und kann von der Klasse „Notebooksimulationframes“ (Kapitel 5.7) verarbeitet werden. Zusätzlich wird ein String deklariert, der den Namen der XML im gesamten System beeinflussen kann („XMLname“).

Die Ordnerstruktur ist ebenfalls ein wichtiger Teil der Anwendung, da sich daraus weitere Funktionen ableiten lassen, wie z.B. die Erweiterbarkeit der Anwendung. Bei Bedarf kann ein Administrator ein weiteres übergeordnetes Modell, z.B. ein Schiff, integrieren und die

Ordnerstruktur von MIWa, speziell "Plane", übernehmen, wie in Abbildung 18 dargestellt. Aus diesem Grund enthält die Klasse unter anderem eine Liste der vorhandenen Ordnernamen, die zur Erstellung eines Menüs (siehe Kapitel 5.5) verwendet werden.

Der Ordner "Plane" enthält derzeit drei weitere Unterordner. Der erste ist "AppData", der Informationen wie die initiale XML-Datei, die Abfolge der Modelle, die Anfangsparameter und einen weiteren Ordner für externe Datenbanken enthält. Diese Dateien werden in späteren Kapiteln näher erläutert.

Darüber hinaus gibt es zwei weitere Ordner, in denen die Submodelle in Form von „.mdzip“ MCSE-Dateien hinterlegt sind. Diese Ordner enthalten auch zusätzliche Dateien wie Matlab-, Python- und Simulink-Dateien, die zu dem jeweiligen Submodell gehören. Ein Beispiel hierfür ist der Ordner „Fuel Cell“, der sich im übergeordneten Ordner „power_system“ befindet.

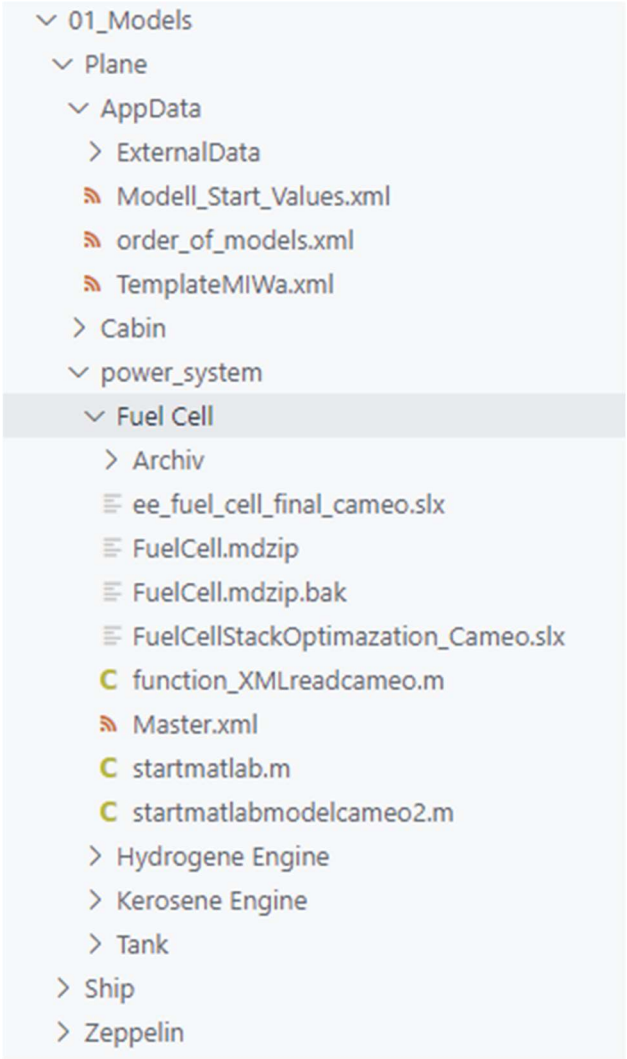


Abbildung 18: Struktur des Ordners 01_Models.

Um zukünftige Submodelle zu einem bestehenden Modell hinzuzufügen, genügt es, einen Ordner mit dem entsprechenden Namen zu erstellen. In diesen Ordner werden alle zusätzlich benötigten Daten, wie z.B. Python- oder Matlab-Extraskripte, sowie die entsprechende MCSE-Datei abgelegt. Außerdem ist es notwendig, die Deklarationen in den Dateien

„order_of_models.xml“ und „Model_Start_Values.xml“ vorzunehmen. Weitere Informationen über diese Dateien werden im Kapitel 5.8 und Kapitel 5.10 bereitgestellt.

5.2 Klasse InfoFrame

„InfoFrame“ ist eine der kürzesten Klassen in der Anwendung, aber weitere Funktionen können hinzugefügt werden. Diese Klasse dient als alternative Kommunikationsmöglichkeit mit dem Benutzer und stellt einen Rahmen (Frame) des externen Moduls „ttkbootstrap“ dar. Das Vererbt die Möglichkeit, konstante Werte immer verfügbar zu machen. Dazu gehört z.B. ein „ttk.bootstrap“ Label, das kontinuierlich verwendet werden kann, um neue Informationen zu veröffentlichen. Die Implementierung dieser Klasse stammt aus vielen kommerziellen Anwendungen, die eine Leiste am unteren Rand des Anwendungslayouts zur Verfügung stellen, z.B. Seitenzahl in der Fußzeile von MS Word-Dokumenten

Um die Nachricht der Anwendung zu aktualisieren, wird die Funktion „updateMessage()“ verwendet, wobei die Nachricht auch innerhalb der Klammern gesendet wird. Es ist wichtig zu erwähnen, dass es zur Vermeidung von Fehlern vorzuziehen ist, immer String-Variablen zu senden. Eine Beispielfunktion zeigt Abbildung 19 mit der Klasse „Menubar“ (siehe Kapitel 5.5). In dieser Funktion ist die Variable "Parent" vorhanden, die auf die Klasse "App" verweist und somit die Verbindung zur Klasse "Infoframe" ermöglicht, um Meldungen anzuzeigen. Ein beispielhaftes Szenario wäre die Meldung, wenn ein neues Modell ausgewählt wurde (Zeile 170). Wie in diesem Fall gibt es mehrere Meldungen in der Anwendung, um den Benutzer auf dem aktuellen Stand zu halten. Die Klasse kann erweitert werden, um neue Funktionen wie Buttons oder andere Widgets zu integrieren.

```
136 class Menubar(Menu):
137 >   def __init__(self,parent):···
166
167     def modelSelection(self,parent,model):
168         parent._currentmodelfolder.set(f"01_Models\\{model}")
169
170         parent.infoframe.updateMessage(f"model {model} selected")
171
172         with open(f"{parent._currentmodelfolder.get()}\AppData\order_of_models.xml") as fd:
173             doc = xmltodict.parse(fd.read())
174         root= doc.keys()
175         root= list(root)
176         doc=doc[root[0]]
177
```

Abbildung 19: Abschnitt der Klasse Menubar.

5.3 Klasse TextBox

Die Klasse „TexBox“ ist ein erweitertes Objekt der Klasse „ttkbootstrap.ScrolledText“, in der der Benutzer seine eigenen Texte hinzufügen kann. In der Praxis nutzt die Anwendung die Möglichkeit, eigene Meldungen hinzuzufügen und ist für den Programmierer die Hauptform der Kommunikation zwischen der Anwendung und dem Benutzer. Eine der erweiterten Funktionen der Klasse ist die Möglichkeit, die veröffentlichten Nachrichten in einer .log-Datei zu speichern. Dies ist nützlich, um aufgetretene Fehler bei der Verwendung der Anwendung nachzuvollziehen, ohne die Anwendung öffnen zu müssen. Abbildung 20 zeigt einen Ausschnitt aus der Datei „Login.log“, in der alle Meldungen mit Zeitstempel gespeichert werden.

```
268  <11/03/2024 13:46:41:: Welcome to MiWa Editor>
269  <11/03/2024 13:46:41:: the parent model as default is the Plane model>
270  <11/03/2024 13:46:41:: Notebooks created successfully>
271  <11/03/2024 13:46:41:: Simulation Notebook created successfully>
```

Abbildung 20: Abschnitt der Login.log Datei.

Diese Klasse beinhaltet bisher eine Funktionsdefinition namens "update_textbox“, die es ermöglicht, Meldungen anzuzeigen und automatisch einen Zeitstempel hinzuzufügen. Von den zwei Eingaben, die diese Funktion benötigt, muss nur die Nachricht selbst angegeben werden. Dabei ist es dem Administrator oder dem Programmierer überlassen, ob diese Nachricht vom Benutzer editiert werden kann, indem zusätzlich ein „True“ oder „False“ übergibt. Ein Beispiel für die Verwendung dieser Klasse durch den Programmierer unterscheidet sich stark vom vorherigen Kapitel, in dem die Verwendung dieser Funktion durch die Verwendung von Vererbung ermöglicht wurde, nämlich „parent.textbox.update_textbox("Hello Word")“.

Es ist geplant, zukünftig weitere Funktionen hinzuzufügen, beispielsweise eine Funktion zur Änderung der Textfarbe, um Fehlermeldungen von normalen Meldungen zu differenzieren. Aus Zeitgründen konnte diese Erweiterung jedoch noch nicht implementiert werden.

5.4 Klasse JsonTool

Die Klasse „JsonTool“ ist eine der wenigen eigenständigen Klassen in dieser Anwendung, die keine Objekte erbt. Die Funktion liegt mehr im Hintergrund als andere Teile der Anwendung. Diese Klasse ist für die langfristige Speicherung von Informationen zuständig. Dazu wird eine „json“-Datei mit der Syntax dieses Dateityps verwendet.

Der Einstieg in das Anwendungsmanagement wird durch die Funktionen des Objekts wesentlich erleichtert. Ein tiefes Verständnis der Funktionsweise von JSON-Dateien ist nicht erforderlich. Die erste Funktion, „initChek()“, ist eine private Eigenschaft, die das Vorhandensein der Datei prüft und diese gegebenenfalls erzeugt. Die beiden anderen Funktionen sind öffentlich und extrahieren und ergänzen Informationen aus der Datei.

Die Funktion „getData“ ist einfach und benötigt nur einen Eingabeparameter, einen Header. Der Header ähnelt einem Tag in einer XML-Datei und enthält entsprechende Informationen. Ein Beispiel wäre die Abfrage des zuletzt von der Anwendung geöffneten Ordners. Mit „Output = parent.jsontool.getData("lastFolder")“ wird nach dieser Information gesucht. wird nach dieser Information gesucht und, falls vorhanden, in der Variablen „Output“ gespeichert. Wenn die Information nicht in der Datei vorhanden ist, wird „False“ zurückgegeben.

Die Funktion „addData“ ist das Gegenstück zur vorherigen Funktion, da sie weitere Informationen hinzufügt. Diese Funktion benötigt zwei Eingaben, einmal den bereits bekannten Header und zugehörige Informationen, zum Beispiel: „parent.jsonTool.addData("lastFolder", "C:/Users/User/Desktop")“. Ist der Header in der Datei nicht vorhanden, wird dieser trotzdem angehängt.

5.5 Klasse MenuBar

Das Objekt „Menubar“ ist, wie die meisten Klassen in der Anwendung, eine Variante einer tkinter-Klasse („tk.Menu“). Mit diesem Objekt kann unter anderem das Simulationsmodell ausgewählt werden (In MIWa ist das Objekt ausnahmlos ein Flugzeug („Plane“)), wie in Abbildung 21 zu sehen ist. An dieser Stelle wird die Thematik der Ordnerstruktur aus Kapitel 5.1 relevant, da durch Anpassung der Ordner und Dateien von MIWa („Plane“) in einem anderen Modell die GUI erweitert werden kann, um z.B. die Vordimensionierung eines Schiffes zu generieren.

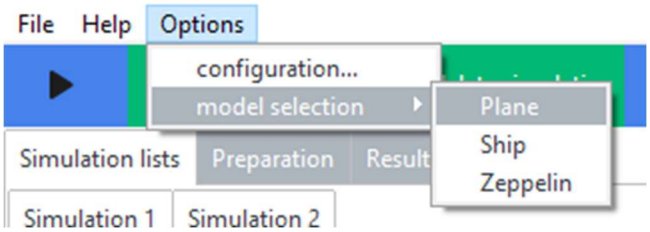


Abbildung 21: Menüleiste für die Modellauswhal.

Dieser Teil des Codes verweist auf das eigene Menu ("self"), welsches die anderen Untermenüs wie "File" enthält. Dies ist in Abbildung 22, Zeile 156 gezeigt. Hier wird zunächst ein Menü mit dem Namen „options“ deklariert. Danach kann mit der Definition "add_command" eine Schaltfläche mit einer Funktion im Menü angezeigt werden, wie in Zeile 157 gezeigt. Die Funktion hat zwei Eingänge, „label“, für die Titel Deklarierung wird, und „command“, wo Funktionen der GUI aufgerufen werden können, wenn der Benutzer sie auswählt, daher der Zusatz „lambda“ bei der Deklaration aller Commands, ansonsten wird die Funktion allein aufgerufen, wenn die gesamte Klasse ausgeführt wird. In diesem Fall hat die GUI-Konfigurationsschaltfläche keine Auswirkung auf die GUI, da die Funktion, die mit ihr verknüpft ist, im Grunde nichts tut, aber wenn die Anwendung erweitert wird, kann Zeile 157 erweitert werden und eine Funktion enthalten, die z. B. ein anderes Fenster mit einem Konfigurationsportal aufruft.

```
155  #options
156  options = Menu(self, tearoff=0)
157  options.add_command(label="configuration...", command=lambda: self.donothing(parent))
158  submenuModels = Menu(self, tearoff=0)
159
160  for model in parent.modelslist:
161
162      submenuModels.add_command(label=model, command=lambda m=model: self.modelSelection(parent, m))
163
164  options.add_cascade(label="model selection",menu= submenuModels)
165  self.add_cascade(label="Options", menu=options)
```

Abbildung 22: Erstellung eines Teils der Menüleiste.

Aus der Eigenschaft „ModelsList“ der Klasse „App (Parent)“ werden in der Schleife in Zeile 160 weitere Möglichkeiten zur Modellauswahl generiert, die auf die Funktion „modelSelection“ verweisen. Zur anschaulichen Darstellung von Funktionen eignet sich ein UML-Tool, das Aktivitätsdiagramm. Dieses Diagramm wird verwendet, um Abläufe wie Algorithmen oder Operationen zu beschreiben. Das Aktivitätsdiagramm wird genutzt, um Abläufe wie Algorithmen oder Operationen zu beschreiben, indem der zu beschreibender Prozess in kleinere Aktionen zerlegt wird [35, S. 244-268].

Wie in Abbildung 24 dargestellt, beginnt der Funktionsablauf mit der Änderung einer Variablen in der Hauptklasse. Danach wird angegeben, welches Modell ausgewählt wird. Danach wird geprüft, ob die Datei „order_of_models.xml“ vorhanden ist. Ist dies nicht der Fall, wird eine Fehlermeldung ausgegeben und die Funktion abgebrochen. Wenn die Datei vorhanden ist, wird diese gelesen und die Namen aller verfügbaren Simulationsmodelle werden in der Hauptklasse gespeichert.

```
<root>
  <sequences>
    <infoformodels>
      Welcome to the aircraft customization interface for the A320 passenger
planes! Select the type of Engine you want your model to have
    </infoformodels>
    <sequence>
      <name>Hydrogen-Based Power System</name>
      <description>The hydrogen-based power system is an innovative and
sustainable solution for aircraft power generation.</description>
      <model>/power_system/Hydrogen Engine</model>
      <model>/power_system/Fuel Cell</model>
      <model>/power_system/Tank</model>
      <model>/Cabin</model>
    </sequence>
  </sequences>
  <info>This is a  file to determine the order of models in the GUI</info>
</root>
```

Abbildung 23: Vereinfachte Struktur "order_of_models.xml".

Die Datei „order_of_models.xml“ ist essentiell für den Ablauf, da sie Informationen über die Simulation und die Submodelle enthält. Zur besseren Veranschaulichung der Dateistruktur wird auf

Abbildung 23 verwiesen. Das XML-Dokument beginnt nach dem Root-Element mit dem Haupttag „sequences“. Diese enthält einen weiteren Tag namens „infoformodels“, der Informationen für die spätere Verwendung in der Anwendung speichert. Zusätzlich können beliebig viele weitere Sequenzen hinzugefügt werden, wie bei der ersten Sequenz „Hydrogen-Based Power System“. Die Funktion „modelselection“ filtert und speichert diese Sequenznamen. In der Abbildung 23 ist zu erkennen, dass zunächst das Wasserstofftriebwerks, anschließend das Brennstoffzellenmodell, danach das Tankmodell und schließlich die Kabine simuliert werden.

Zurzeit kann diese Datei nur mit einem externen Texteditor bearbeitet werden. Der vorliegende Code enthält jedoch bereits Strukturen, die es einer Klasse ermöglichen würden, weitere Sequenzen oder Submodelle hinzuzufügen oder zu bearbeiten.

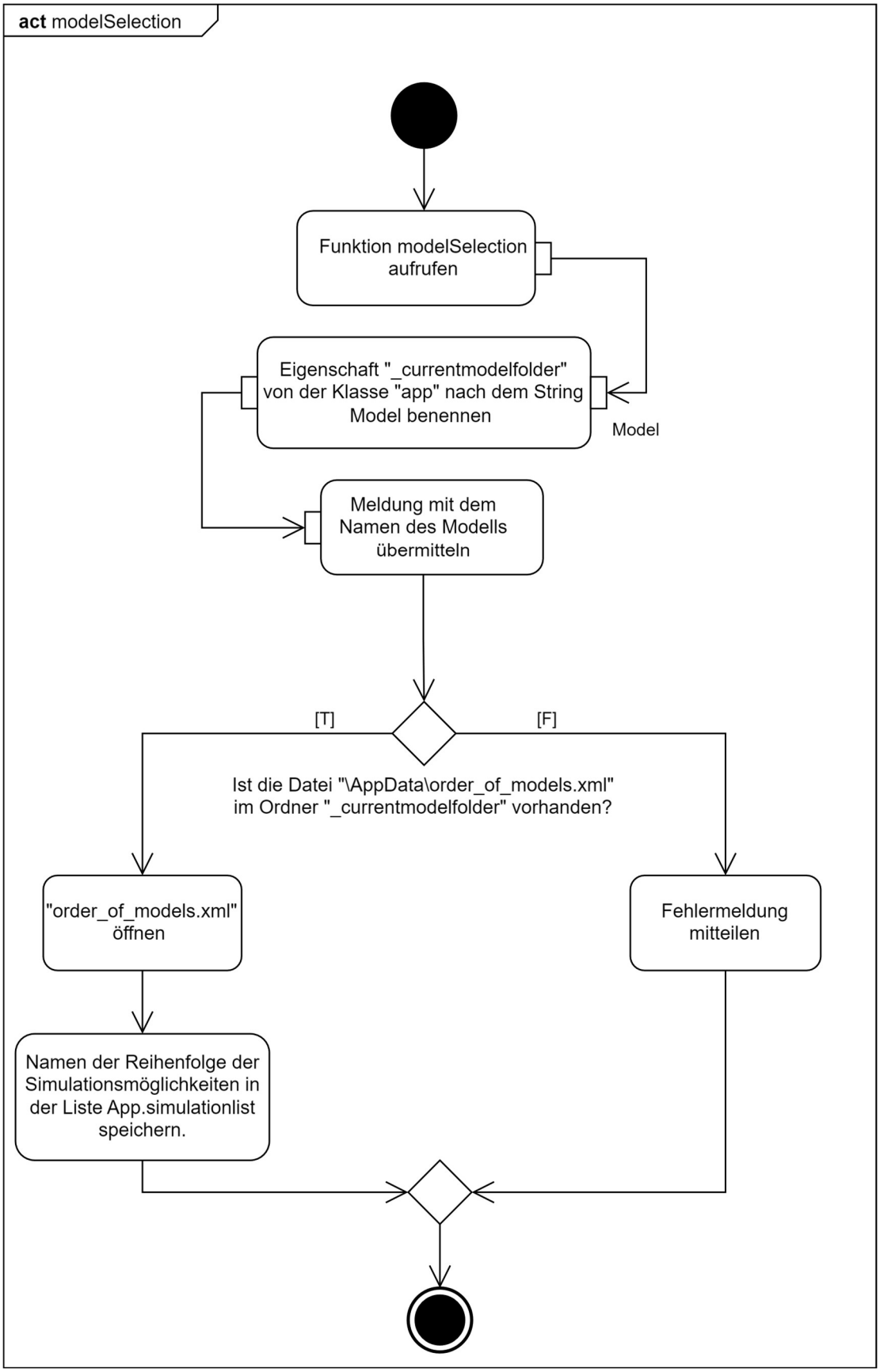


Abbildung 24: Aktivitätsdiagramm der Definition "modelSelection".

5.6 Klasse Notebooks

Das „Notebook-Objekt“ ist eine Ableitung der Klasse „ttkbootstrap.Notebook“ und verwaltet die verschiedenen Fenster der grafischen Benutzeroberfläche, wie in Abbildung 25 dargestellt. Für die Erstellung von Notizbüchern ist es notwendig, einen neuen Frame zu erzeugen und dann die Methode „add“ anstelle der tkinter-Methode „pack“ zu verwenden. Die Methode „add“ wurde speziell für die Verwendung mit Notebooks und Frames entwickelt, wie in Abbildung 24 Zeile 483 dargestellt.

```
471 class Notebook(ttk.Notebook):
472     def __init__(self,parent):
473         super().__init__(parent)
474         self.config(bootstyle="secondary")
475         simulationframes= ttk.Frame(self)
476         tabPreparation = ttk.Frame(self)
477         tabResults = ttk.Frame(self)
478         tabPDFexport = ttk.Frame(self)
479 
480         #
481         parent.textbox.update_textbox("Notebooks created successfully")
482         #
483         self.add(simulationframes, text="Simulation lists")
484         self.add(tabPreparation, text="Preparation")
485         self.add(tabResults, text="Results")
486         self.add(tabPDFexport, text="PDF export")
487 
488         self.pack(side="bottom", fill="both", expand=True)
489 
490         #simulation layout
491         self.notebooksimulationframes = self.Notebooksimulationframes(simulationframes,parent)
```

Abbildung 25: Abschnitt der Kasse Notebook.

Trotz der Simplizität der Klasse besteht die Möglichkeit der Erweiterung duch die Integration weitere Notebooks und Widgets. Derzeit liegt der Fokus hauptsächlich auf dem „Simulation list“-Framework, wie in Abbildung 26 dargestellt und Kapitel 5.7 beschrieben. Erste Ansätze für neue Aktionen und Funktionalitäten sind jedoch erkennbar. Zum Beispiel der "Results"-Reiter, der darauf abzielt. Die finale Simulations-XML mit anderen zu vergleichen, oder der PDF-Export, der die Simulationsergebnisse in eine lesbare Form umwandeln soll. Diese Möglichkeiten befinden sich noch in der Entwicklungsphase und sind in den ursprünglichen Anwendungsfällen nicht beschrieben.

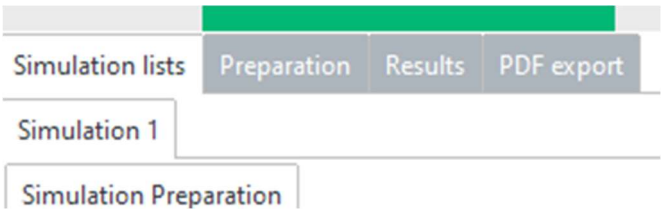


Abbildung 26: Kassenlayout Notebook.

5.7 Kasse Notebooksimulationframes

„Notebooksimulationframes“ ist eine Ableitung der Klasse ttkbootstrap.Notebook, wie die vorherige Klasse. Diese Klasse ist eine Unterklasse der in Kapitel 5.6 beschriebenen Klasse. Daher wird beim Zugriff auf die Eigenschaften der zentralen Klasse „app“ (Kapitel 5.1) der Anfang „grandfather“ anstelle von „parent“ verwendet. Die Klasse verwaltet die verschiedenen Simulationen, die erzeugt werden können (siehe Abbildung 27), indem eine Liste der verschiedenen Simulationen und ihrer Eigenschaften erzeugt wird, mittels der Eigenschaft „self.ListSimConfigs“.

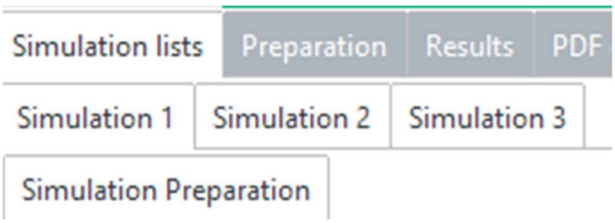


Abbildung 27: Klassenlayout Notebooksimulationframes.

Damit die Oberklasse, „app“ (siehe Kapitel 5.1) weiß, wie viele Simulationen vorhanden sind, wird im Konstruktor der Klasse Notebooksimulationframes die gesamte Klasse in der Liste „SimulationNotebooksList“ der Oberklasse „app“ gespeichert. Dadurch können in Zukunft andere Klassen, wie z.B. die Klasse „SimulationFrame“ (Kapitel 5.11), leichter auf diese Information zugreifen. Für das initiale Layout der Simulation ist die Subklasse “SelectonbarForSim“ (siehe Kapitel 5.8) zuständig. Der Aufruf dieser Klasse wird in der bereits erwähnten Eigenschaft, „ListSimConfigs“, gespeichert.

Die letzte Eigenschaft dieser Klasse ist ein Zähler, der in direktem Zusammenhang mit den beiden Funktionen der Klasse steht. Diese zählt die Anzahl der Simulationen. Die erste Funktion heißt addNotebook und erzeugt , eine neue Simulation. Die zweite Funktion delateNotebook ist für das Löschen von zusätzlichen Simulationen zuständig. Da diese Definitionen in direktem Zusammenhang mit dem Kapitel 5.11 stehen, wird die Funktion in diesem Kapitel näher erläutert.

5.8 Klasse SelectonbarForSim

Das Objekt verhält sich wie ein angepasster „ttkbootsrap-Frame“, da diese Klasse immer aufgerufen wird, wenn eine neue Simulation erstellt wird. Die Hauptfunktion der Klasse bietet dem Benutzer die Möglichkeit, eine Simulationssequenz auszuwählen. Diese Sequenzen werden in der Datei „order_of_models.xml“ vom Administrator gespeichert (siehe Kapitel 5.5). Abbildung 28 zeigt ein Beispiel verschiedenen Simulationsmöglichkeiten.

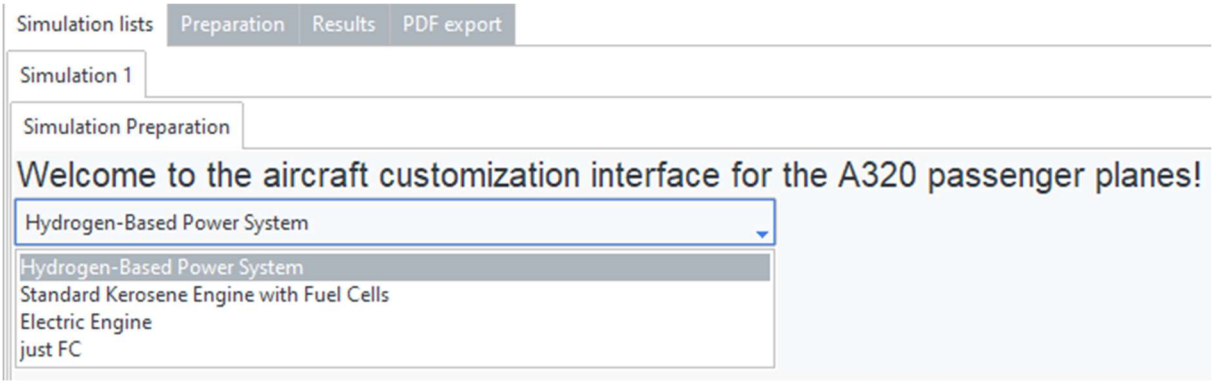


Abbildung 28: Klassenlayout SelectonbarForSim.

Das Aktivitätsdiagramm in Abbildung 29 zeigt die Reihenfolge der Aufgaben der Klasse. Zuerst werden die Eigenschaften deklariert, wobei die meisten als leere Listen definiert sind, bspw. „VariablenList1“. Dann wird „VariablenList1“ mit den Namen der Simulationssequenzen gefüllt. Diese Sequenzen sind bereits aus der XML-Datei in die Klasse „Menubar“ (Kapitel 5.5) importiert. Anschließend wird das „Simulationsnotizbuch“, „Vorbereitung“, platziert. Danach wird nach einem „Tag“ namens „infoformodels“ gesucht. Bei erfolgreicher Suche wird der Text importiert, ansonsten wird ein Standardtext verwendet. Mit dem Text wird ein „Label“ erstellt und darunter ein „Drop-Down“-Menü mit den Optionen aus „VariablenList1“ platziert. Wenn der Benutzer eine Option auswählt, wird die Klasse Subframes aufgerufen. Diese Klasse erstellt zusammenfassend personalisierte Frames, die in der Liste „Subframes“ gespeichert werden.

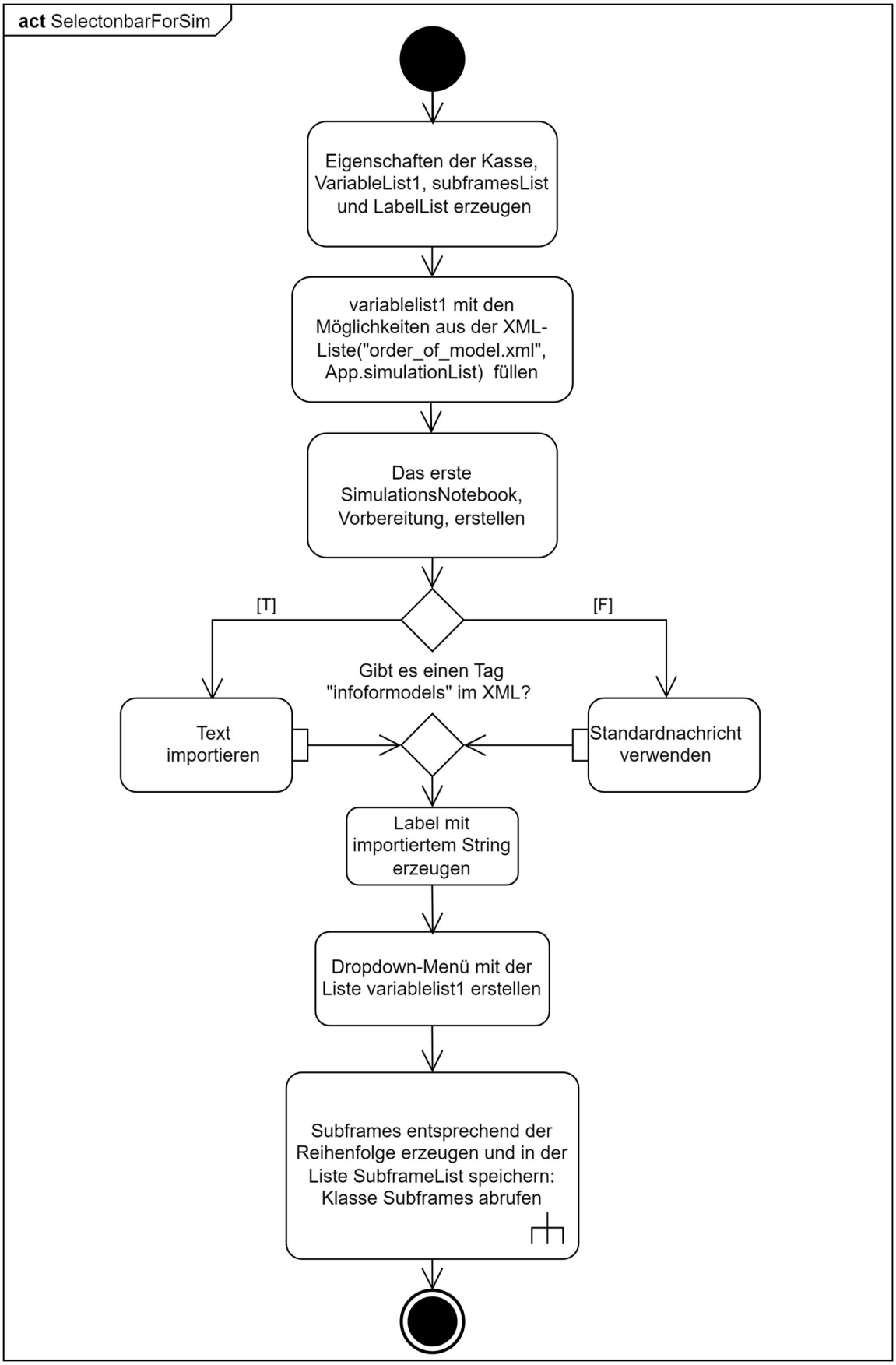


Abbildung 29: Aktivitätsdiagramm der Klasse SelectonbarForSim.

5.9 Klasse SubFrames

Diese Klasse erbt keine weiteren Eigenschaften von anderen Modulen. Die Hauptfunktion ist die Erzeugung mehrerer Frames, damit die einzelnen Modelle der gesamten Simulation entsprechend den Anforderungen des Benutzers angepasst werden können. Dies ermöglicht dem Benutzer die Eingabe individuelle Startparameter. Ein Beispiel ist in Abbildung 30 dargestellt, die den ersten Simulationslauf zeigt.

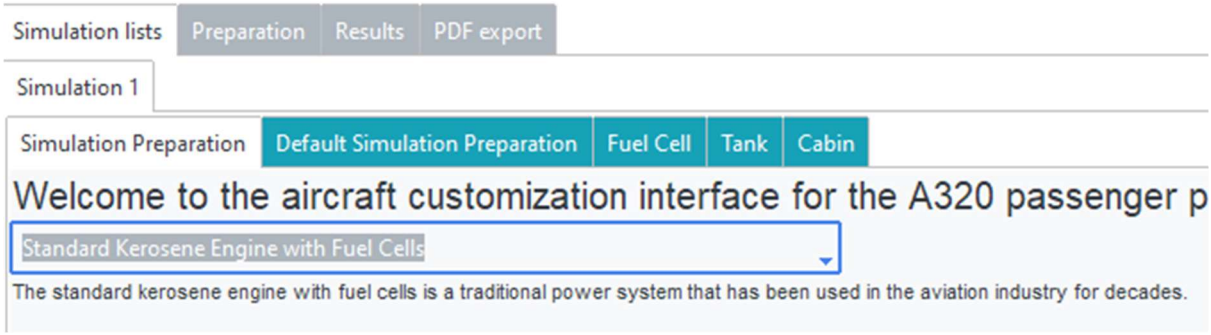


Abbildung 30: Klassenlayout SubFrames.

Die Verwaltung der einzelnen Frames beginnt mit der Deklaration von drei Listen. Die erste ist „ScrolledFrames“, wo spezielle Frames mit der Möglichkeit, im Frame nach unten zu scrollen, wenn zu viele Informationen angezeigt werden, diese Eigenschaft ist sehr nützlich für Modelle mit vielen Anfangsparametern. „Listofmodels“ ist die zweite angegebene Liste. Diese Liste ist für die Klasse „SimulatioFrame“ relevant, da eine XML-Datei mit den Startparametern in verschiedene Ordner verschoben werden muss (siehe Kapitel 2.11). „Models“ ist eine Vereinfachung der vorherigen Liste, da sie nur den Namen eines Modells enthält, und wird für die Benennung der verschiedenen Reiter verwendet, die wiederum in Abbildung 30 zu sehen sind.

Anschließend wird überprüft, ob die Liste Framelist aus Kapitel 5.8 nicht bereits leer ist. Wenn dies der Fall ist, wird bestätigt, dass die Funktion für die Simulation nicht zum ersten Mal aufgerufen wird und löscht die alten Subframes, um neue zu erzeugen. Wenn keine alten Subframes vorhanden sind, kann dieser Teil ignoriert werden.

Als nächstes werden die Listen „Listofmodels“ und „Models“ mit den Informationen über die Modelle aus der Datei „order_of_models.xml“ gefüllt. Schließlich werden die verschiedenen Frames durch eine Schleife erzeugt und zusätzlicher Frame wird erstellt, in den die Standardwerte der Simulation eingefügt werden können. Dieses Verfahren wird in Kapitel 5.10 näher erläutert. Der beschriebene Ablauf ist in Abbildung 31 dargestellt.

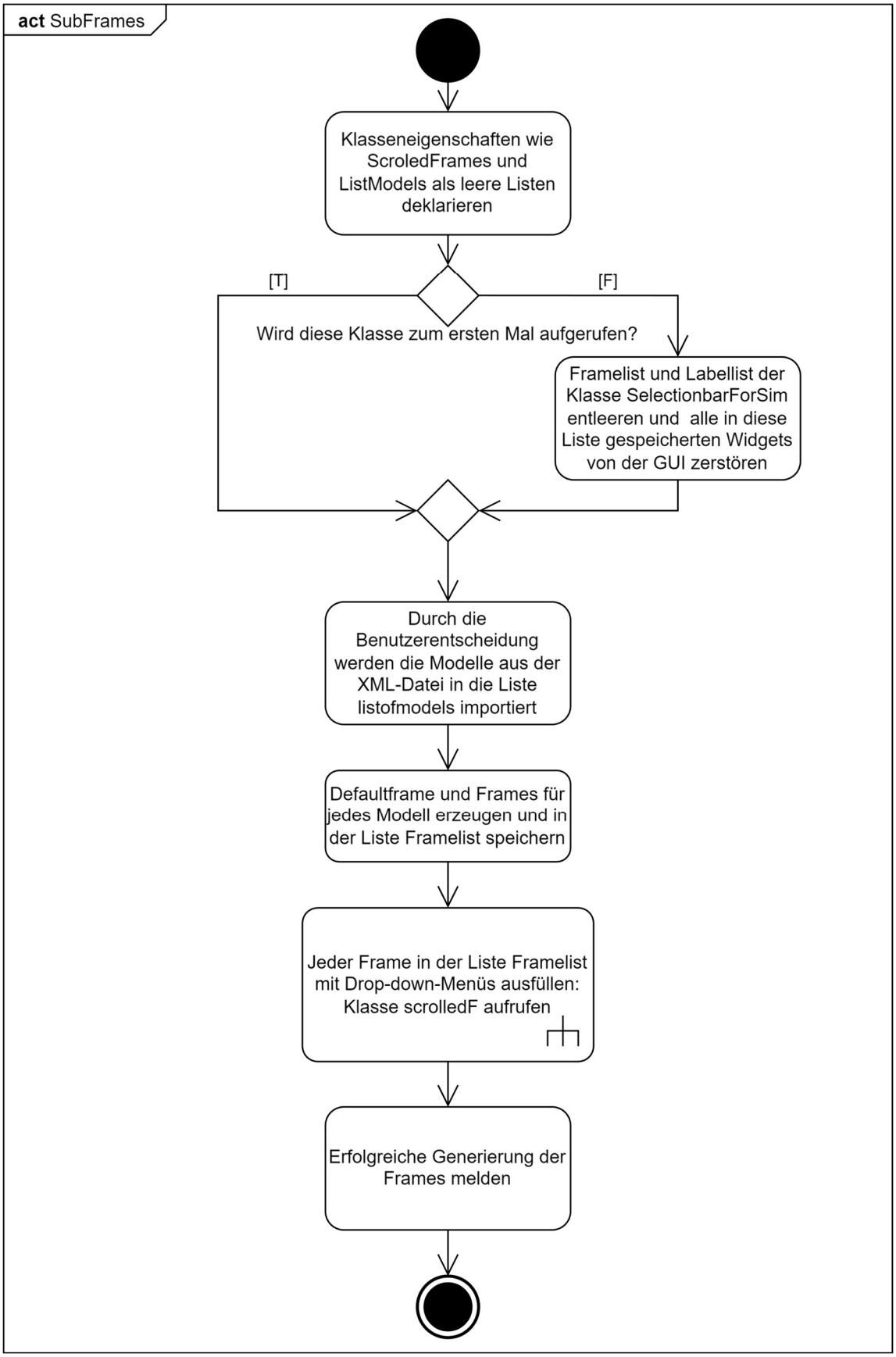


Abbildung 31: Aktivitätsdiagramm der Klasse SubFrames.

5.10 Klasse scroledF

Diese Klasse ist verantwortlich für die Generierung der Eingabemöglichkeiten in den einzelnen Subframes. Die Art und Weise, wie die Eingabeoptionen in der Applikation generiert werden, wird durch die externe Speicherung der Startparameter in einer XML-Datei bestimmt. Diese Datei heißt „Model_Start_Values.xml“ und befindet sich im Ordner „AppData“ der Ordnerstruktur. Der Dateiabschnitt in Abbildung 32 zeigt die Startparameter für ein Modell. Ein Beispiel ist der Sicherheitsfaktor des Tanks und die verschiedenen zusätzlichen Optionen, die der Benutzer auswählen kann.

```
<root>
  <Models>
    <Model><name>Tank</name>
      <parameters>
        </parameter>
        <parameter>
          <name>SafetyFactor</name>
          <description>Tank safety factor</description>
          <value>1.2</value>
          <value>2</value>
          <value>1.5</value>
        </parameter>
        <externlgroup>
          <name>HydrogenTankAlloy</name>
          <path>\AppData\ExternalData\External_DB_Materials_metals.csv</path>
          <columnnames>Material</columnnames>
          <parameter>
              <name>rpInnerWall</name>
              <description>Offset Yield Point 0.2% off inner wall material  [MPa]</description>
              <column>Sy</column>
          </parameter>
          <parameter>
              <name>rpOuterWall</name>
              <description>Offset Yield Point 0.2% off outer wall material  [MPa]</description>
              <column>Sy</column>
          </parameter>
        </externlgroup>
        <parametergroup>
          <name>Insolation</name>
          <parameter>
            <name>ThermalConductivityInsulation</name>
```

Abbildung 32: Abschnitt aus „Models_Start_Values.xml“.

Wie in Abbildung 33 zu sehen ist, wird bei der Ausführung der Auswertung automatisch eine Reihe von Dropdown-Menüs angezeigt, die die Optionen aus der XML-Datei enthalten. Außerdem sind diese Variablen mit der Adresse versehen, die in der SimulationXML-Datei gespeichert ist. Damit kann die Funktion "Playsimulation" der Klasse „SimulationFrame“ (siehe Kapitel 5.11) auf die Variablen zugreifen. Am oberen Ende des Layouts ist das erste Dropdown-Menü zu sehen, in dem der Benutzer zwischen Standard- und Expert-User wählen kann. Mit der Standard-Option ist es nur möglich, die vom Administrator zur Verfügung gestellten Werte zu verwenden. Der "Expert"-Modus bietet dem Benutzer die Möglichkeit, individuelle Startwerte einzugeben.

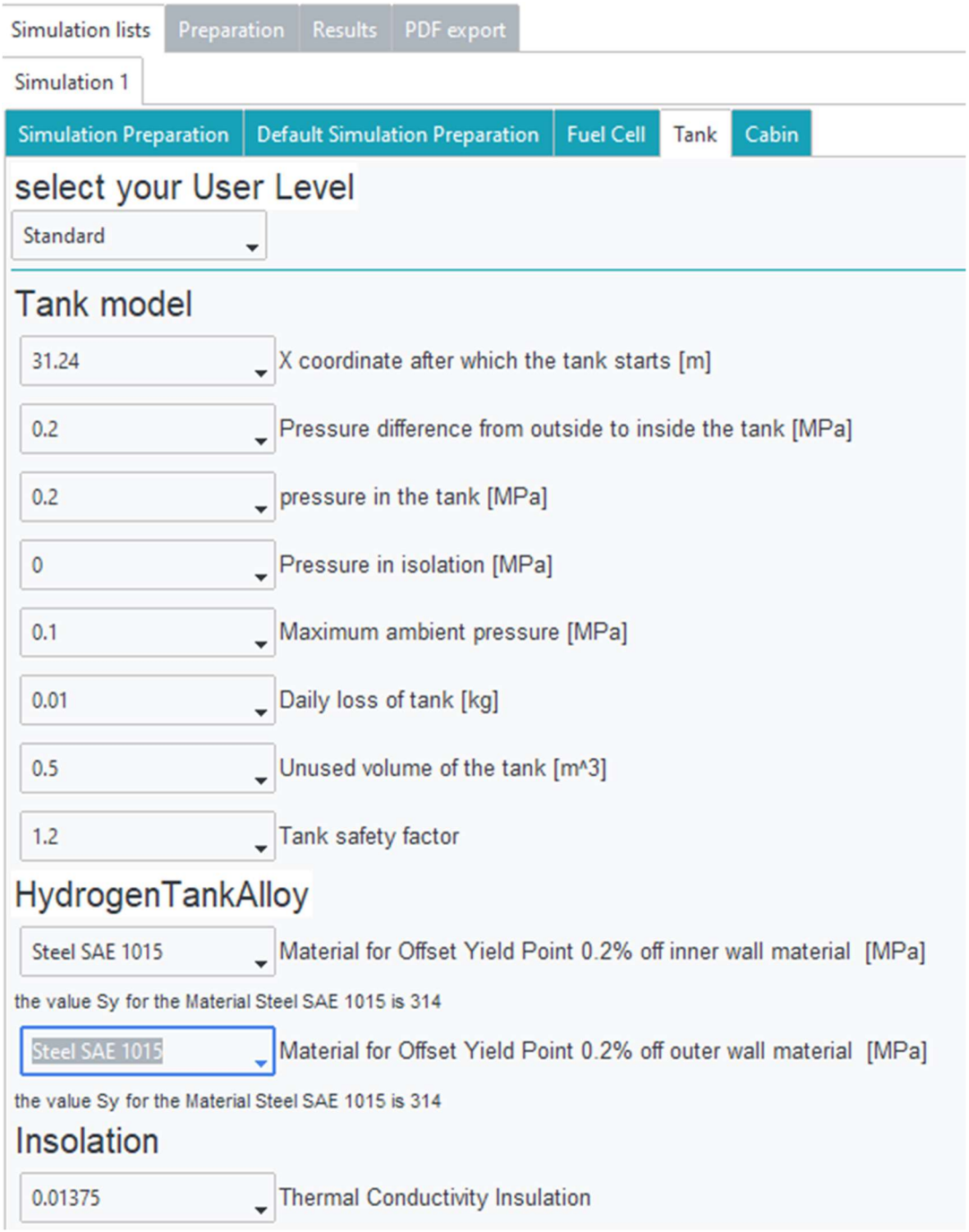


Abbildung 33: Klassenlayout scroledF.

Eine weitere Funktion dieses Moduls ist die Verwendung externer Datenbanken. Der Abschnitt in Abbildung 32 ermöglicht die Eingabe eines Speicherorts der Datenbank und damit die Funktionalitäten und Parameter des Modells zu erweitern.

Der funktionale Teil der Klasse beginnt mit der Reihenfolge der Eigenschaften, wie z.B.: ein Boolean mit dem Namen „readytosim“, der auf „False“ gesetzt wird. Dazu können der Modellname und weitere Listen, in denen die verschiedenen Variablen, Menüs, Labels und Frames gespeichert werden.

Anschließend werden die Informationen aus der XML-Datei, in der die Startparameter gespeichert sind, geöffnet und nach dem Modellnamen gefiltert, z.B. die Informationen aus dem Modell „Tank“(siehe Abbildung 32). In den die zwei genannten Optionen "Standard" und "Expert" zur Verfügung stehen. Das erste Menü hat auch die Fähigkeit, zu erkennen, wenn es nicht zum ersten Mal verwendet wird, um unerwünschte alte Widgets zu löschen.

Wenn eine Option ausgewählt wird, beginnt eine Schleife, die drei Fälle unterscheidet. Der erste Fall ist ein einfacher Parameter, bei dem zuerst ein Rahmen erzeugt und in einer Liste gespeichert wird. Ein Titel wird in den Rahmen gesetzt und ein Menü wird aus den in der XML-Datei gespeicherten Daten generiert. Der Titel wird ebenfalls der XML-Datei entnommen. Der zweite Fall ist eine Gruppe von Parametern, die z.B. einen übergeordneten Begriff haben. Dieser Fall ist dem ersten ähnlich, mit dem Unterschied, dass vor der Erzeugung des Rahmens ein großer Titel mit dem Namen des übergeordneten Begriffs erzeugt wird. Diese Situation ist unten in Abbildung 33 zu sehen. Die „Isolation“ ist der Oberbegriff und die Eigenschaften des Materials folgen. Der dritte und letzte Fall betrifft die Verwendung externer Datenbaken, die im CSV- oder XLSX-Format deklariert werden können, um die Auswahlmöglichkeiten erheblich zu erweitern. Wie im zweiten Fall wird ein Oberbegriff in einem Titel verwendet und anschließend die Menüs mit den zusätzlichen Informationen generiert. Im Modell „Tank“ wird diese Möglichkeit angeboten, damit die Streckgrenze, eine Materialeigenschaft, ausgewählt werden kann. Diese letzte Möglichkeit bietet einen großen Vorteil bei der Erweiterung der Anwendung durch die Verwendung externer Datenbanken.

Eine weitere Möglichkeit, die aus Zeitgründen nicht in diese Version der Anwendung integriert werden konnte, ist die Einführung einer Option in der Klasse „MenuBar“ von Kapitel 5.5, um eine Excel-Tabelle zu öffnen, die automatisch in die Datei „Model_Start_Values.xml“ mit Hilfe eine neue Klasse konvertiert wird. Dies ist von Vorteil für alle Benutzer, die Werte bearbeiten möchten. Die letzte Möglichkeit, die in einer zukünftigen Version implementiert werden soll, ist die intelligente Erkennung von Buchstaben oder Symbolen im Expertenmodus, um Eingabefehler zu vermeiden. Diese Möglichkeit erfordert eine Umstrukturierung der Klasse.

5.11 Klasse SimulationFrame

Das Objekt „SimulationFrame“ erhält seine Eigenschaften aus der Klasse „ttkbotsrap.Frame“ und enthält darüber hinaus weitere Buttons, die einen konkreten Bezug zur Klasse „Notebookssimulationsfames“ in Kapitel 5.7 und deren Funktionen herstellen. Abbildung 34 zeigt diese Eigenschaften.

Abbildung 34: Klassenlayout SimulationFrame.

Zuerst wird der Frame mit der gesamten Klasse ganz oben unter dem Menü im Design der Anwendung platziert. Anschließend werden mit dem Modul „Pillow2 die Bilder (die Symbole Play, Plus und Minus, siehe Abbildung 34) importiert und in die Knöpfe integriert. Jede Schaltfläche hat eine Funktion. Die des mittleren Buttons öffnet einen weiteren Rahmen, in dem weitere Module ausgewählt werden können, indem die Funktion „addNotebook“ der Klasse „Notebookssimulationsfames“ (siehe Kapitel 5.7) aufgerufen wird.

Da die Klasse „NotebookSimulationFrames“ einen internen Zähler hat, der die Anzahl der ausgeführten Simulationen zählt, wird direkt nach dem Aufruf der Funktion „addNotebook“ der Zähler um eins erhöht. Anschließend wird geprüft, ob der Zähler größer als eins ist. Wenn dies der Fall ist, wird die Schaltfläche „Simulation löschen“ aktiviert und kann vom Benutzer verwendet werden. Weiterhin wird ein Notizbuch mit dem Namen „Simulation“ und der Nummer des Zählers in der GUI erstellt. Schließlich wird die Klasse „SelectionbarForSim“ aus Kapitel 5.8 in dem neuen Frame, der in das Notebook integriert ist, aufgerufen und in der Liste „ListSimConfigs“ gespeichert. Und damit kann der normale Ablauf der Konfiguration der Simulation, wie beim ersten Modell, gestartet werden. Wenn eine Simulation gelöscht werden soll, wird diese Möglichkeit durch die zweite Schaltfläche angeboten. Diese ruft die zweite Funktion der Klasse „NotebookSimulationFrames“ auf und löscht das unerwünschte Notebook aus den Listen und Abschießend dem Layout der grafischen Benutzeroberfläche.

Die letzte Funktionalität dieser Klasse ist die Funktion „playsimulation“, die sich um die Erstellung der *StartXML-Datei* mit den vom Benutzer gewählten Parametern kümmert. Zusätzlich werden weitere Informationen wie Datum und Versionsnummer in die Datei eingefügt. Zum besseren Verständnis wird die Funktion nachfolgend erläutert. Wenn der Benutzer den Play-Button drückt, erscheint ein neues Fenster mit der Frage, ob die Simulation gestartet werden soll. Ist dies der Fall, wird zunächst wieder darauf hingewiesen, dass Maus und Keyboard nicht verwendet werden dürfen. Danach werden die Tasten der gesamten Klasse deaktiviert, um Fehler zu vermeiden.

Mit Hilfe der Klasse JsonTool wird die XML-Version aus der „config.json“ ausgelesen und in einer Variable gespeichert. Diese Variable wird um eins erhöht und in die Json-Datei zurückgeschrieben. Eine weitere Variable wird erzeugt, in dem der Zeitpunkt des Simulationsstatus gespeichert wird.

Die Klasse „notebooksimulationframes“ enthält eine Liste aller erstellten Simulationen. Aus dieser Liste wird die gesamte Simulationsschleife erstellt. Die Schleife beginnt mit einer Kopie der Muster-XML, Welsche im Ordner „AppData“ hinterlegt ist. In diese Kopie werden extra Informationen gespeichert, bspw. das Datum, die Version im Header wie in Abbildung 35 dargestellt.

```
<MIWa version="1.0" encoding="UTF-8">
  <header>
    <name />
    <date>21/03/2024 10:08:39</date>
    <timestamp>1711012119.6998065</timestamp>
    <version>91.1</version>
    <variant>1</variant>
    <model>Fuel Cell</model>
  </header>
```

Abbildung 35: Header einer XML-Simulationsdatei.

Anschließend wird die angepasste XML-Datei formatiert und in „Master.xml“ umbenannt. Danach beginnt eine zweite Schleife, die auf einer Liste „listofmodels“ basiert, in der die Ordner mit den MCSE-Dateien gespeichert sind. Die Schleife verschiebt die Datei „Master.XML“ in diesen Modellordner und verwendet die Klasse „autostartcameo“ aus Kapitel 5.12, um die Modelle zu starten und zu simulieren. Wenn die Simulation nicht erfolgreich war, erhält der Benutzer eine Meldung. Wenn alles in Ordnung ist, werden die Ergebnisse im Ordner der Ordnerstruktur „04_results“ unter einem Namen mit dem Datum und Uhrzeit gespeichert.

Wenn alle Simulationen durchgeführt wurden, wird gemeldet, dass die Anwendung wieder verwendet werden kann, und der Benutzer wird gefragt, ob die Ergebnisse an einem anderen Ort gespeichert werden sollen. In diesem Fall kann angegeben werden, wo die Datei gespeichert werden soll, und die Funktion wird beendet. Eine zusammengefasste Version des Ablaufs ist im Aktivitätsdiagramm in Abbildung 36 dargestellt.

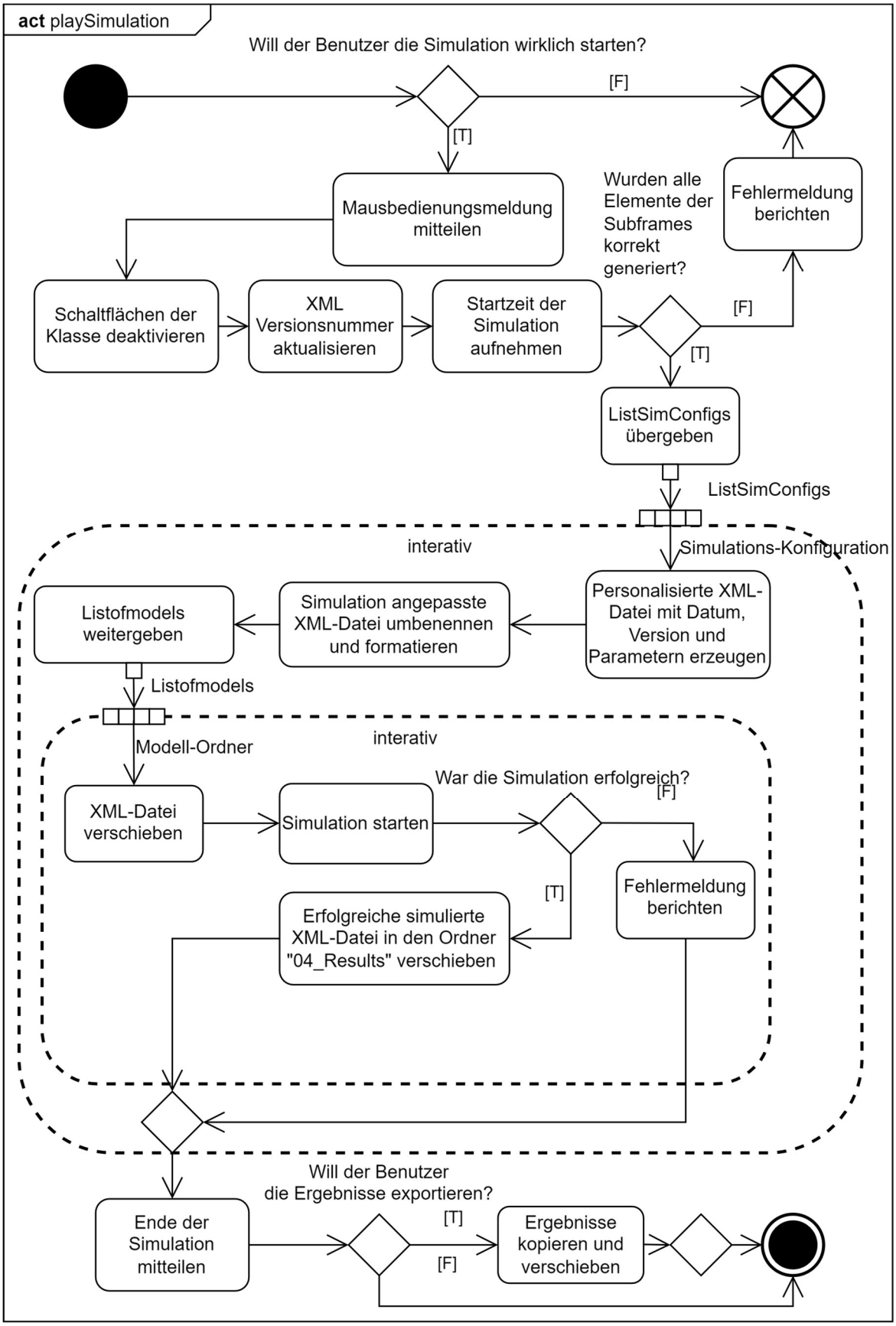


Abbildung 36: Aktivitätsdiagramm der Funktion Playsimulation.

5.12 Klasse autostartCameo

„Autostartcameo“ ist eine selbst entwickelte Klasse mit der Aufgabe, die MCSE-Modelle zu starten und die verschiedenen Konfigurationen auszuführen. Diese Klasse ist für den Erfolg des Projekts von entscheidender Bedeutung, da die Anwendung ansonsten keinen Mehrwert für den Benutzer hätte. Die Entwickler von MCSE bieten in der Anleitung des Programms die Möglichkeit an, die Modelle durch Befehle auszuführen. Diese Methode hat allerdings nicht die Anforderungen erfüllt, so dass der Programmierer einen anderen Weg der Automatisierung wählt. Zum besseren Verständnis der Aufgabe wird zuerst gezeigt, wie ein Mensch eine Simulation ausführen kann.

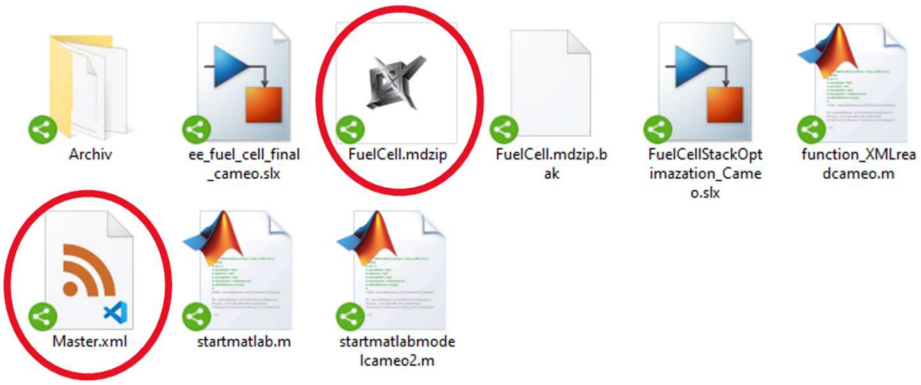


Abbildung 37: Ordnerstruktur für Modellstart

Zuerst muss überprüft werden, ob im Speicherort des auszuführenden Modells (siehe Abbildung 37) eine „.mdzip“- und eine „.xml“- Datei mit dem Namen „Master.xml“ vorhanden sind. Die XML-Datei muss die Struktur und die Anfangsparameter der Simulation enthalten. Normalerweise wird diese Datei vom Systemingenieur manuell erstellt und zur Verfügung gestellt. Außerdem müssen alle zusätzlichen Skripte für die Simulation in diesem Ordner vorhanden sein. Ein Beispiel ist das Brennstoffzellenmodell in Abbildung 37. Wenn alle notwendigen Dateien vorhanden sind, kann die „mdzip“-Datei im MCSE geöffnet werden. Anschließend muss abgewartet werden, bis MCSE erfolgreich gestartet wurde.

Im oberen Teil des Programms befindet sich ein Dropdown-Menü, das für alle Modelle des Projekts nach dem EVA-Prinzip standardisiert wurde, das in Kapitel 2.1 näher erläutert ist. Um die Anfangsparameter in das Programm zu importieren, muss zunächst „1Inport“ im Menü ausgewählt und der Play-Button angeklickt werden. Wenn das Programm meldet, dass der Import erfolgreich war, kann mit „2 Execution“, die zweite vorkonfigurierte Simulation, gestartet werden. Dieser Teil der Simulation ist in der Regel der zeitaufwendigste, da hier rechenintensive Matlab-, Simulink- und Python-Skripte gestartet werden. Anschließend werden die Simulationsergebnisse mit „3 Output“ in die „Master.xml“ exportiert. Die drei Simulationskonfigurationen sind in Abbildung 38 dargestellt.

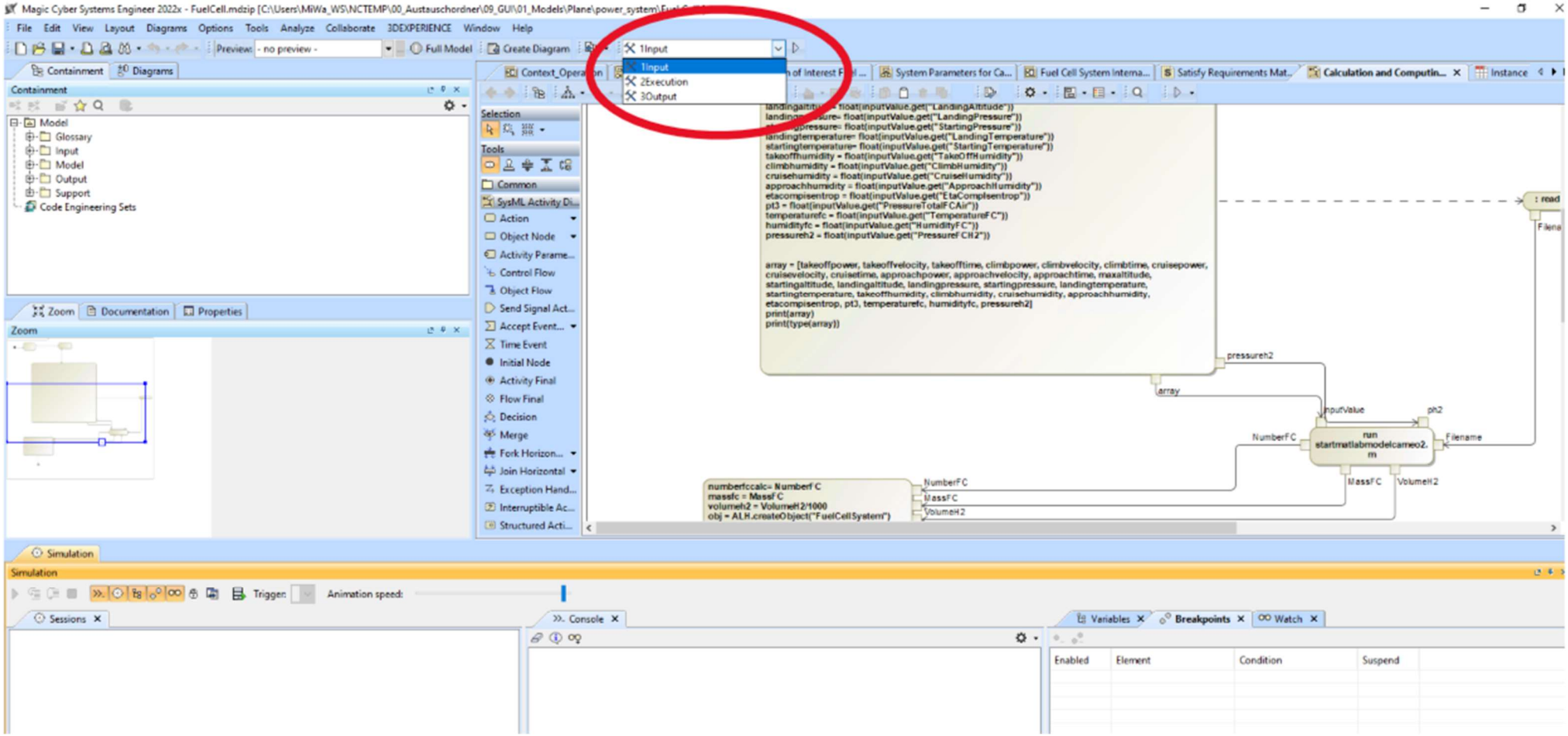


Abbildung 38: Start einer Simulation in MCSE

Nachdem alle beschriebenen Schritte durchgeführt sind, kann die XML-Datei für eine Übersicht der Ergebnisse geöffnet werden. Wenn ein zusätzliches Modell, wie z.B. die Dimensionierung eines Wasserstofftanks, gestartet werden soll, muss der Vorgang im Ordner des Tankmodells wiederholt werden. Offensichtlich ist diese Aufgabe für die verschiedenen Simulationsvarianten umständlich, zeitaufwendig und fehleranfällig. Die Klasse „autostartcameo“ soll hier Abhilfe schaffen.

Bevor die Klasse vorgestellt wird, sollen wichtige zusätzliche Dateien erwähnt werden. Eine davon ist die Datei „simulation.log“. Die „.Log“-Datei dient als Historie für alle Kommentare, die in der MCSE-Konsole veröffentlicht werden. Sie stellt eine wesentliche Grundlage für diese Klasse dar, da MCSE darin vermerkt, wann eine Simulation beendet wurde. Ein Auszug aus der „simulation.log“ ist in Abbildung 39 dargestellt.

simulation.log - Editor

Datei Bearbeiten Format Ansicht Hilfe

```
<log4j:event logger="SIM_USER_OUTPUT_CONSOLE-1" timestamp="1682678046675" level="INFO" thread="SIM-Initialize Solving">
<log4j:message><![CDATA[<html>00:00:00,000 : **** Block  <a href="_2022x_11ad0396_1679321991569_448684_23223">ReadData</a> is initialized. ****</html>]]></log4j:message>
</log4j:event>

<log4j:event logger="SIM_USER_OUTPUT_CONSOLE-1" timestamp="1682678048599" level="INFO" thread="SIM-Thread-4">
<log4j:message><![CDATA[<html>00:00:00,000 : **** Block ReadData  <a href="_2022x_11ad0396_1679321991573_428852_23224">behavior</a> is started! ****</html>]]></log4j:message>
</log4j:event>

<log4j:event logger="SIM_CONSOLE-1" timestamp="1682678051505" level="ERROR" thread="SIM-Thread-3">
<log4j:message><![CDATA[<html>00:00:02,906 ERROR: Cannot evaluate <a href="_2022x_11ad0396_1679321991575_764202_23225">Opaque Behavior readAndGenerateInstanceFromXML</a>:</html>]]></log4
</log4j:event>

<log4j:event logger="SIM_PLAIN_CONSOLE-1" timestamp="1682678051506" level="ERROR" thread="SIM-Thread-3">
<log4j:message><![CDATA[IOError: [Errno 2] No such file or directory: u'D:/GATE_MIWa/01_AGATE/Template_Partialmodelle_Koppeln/MasterXML_AgateV2.xml' in <script> at line number 5
expression body: #read xml
import xml.etree.ElementTree as ET
Filename='D:/GATE_MIWa/01_AGATE/Template_Partialmodelle_Koppeln/MasterXML_AgateV2.xml'
###Filename=os.path.abspath(os.path.dirname(__file__)) Befehl für Anwahl des aktuellen Verzeichnisses, damit Pfad nicht geändert werden muss
tree = ET.parse(Filename)
xml_root = tree.getroot()

#findet nur cabin components
test = xml_root.findall(".//powersystem//component")
print(len(test))
for i in test:
        type_xml = i.findtext("type")
        print(type_xml)
        if type_xml == "keroseneturbofan":
                id_xml = i.findtext("ID")
                print(id_xml)
                obj = ALH.createObject("kerosene_turbofan")
                ALH.setValue(obj,"ID",id_xml)
                ALH.addValue(self,"kerosene_turbofan",obj)]]></log4j:message>
</log4j:event>

<log4j:event logger="SIM_USER_OUTPUT_CONSOLE-1" timestamp="1682678051509" level="INFO" thread="SIM-Thread-3">
<log4j:message><![CDATA[<html>00:00:02,909 : **** Activity  <a href="_2022x_11ad0396_1679321991573_428852_23224">ReadXML</a> execution is terminated. ****</html>]]></log4j:message>
</log4j:event>

<log4j:event logger="SIM_USER_OUTPUT_CONSOLE-1" timestamp="1682678051521" level="INFO" thread="SIM-Thread-3">
```

Zeile 31, Spalte 46 | 100% | Windows (CRLF) | UTF-8

Abbildung 39: Ausschnitt aus der Datei „simulation.log“

Da diese Klasse mit einer Bilderkennungsfunktion des Moduls „pyautogui“ arbeitet, sind Bilder des MCSE-Layouts zur Identifizierung der Programmphase relevant. In Abbildung 40 ist zu erkennen, dass das Programm bereit für die Simulation ist.

Abbildung 40: Abschnitt MCSE Layout.

Nachdem alle externen Dateien deklariert wurden, kann mit der Ausführung der Klasse begonnen werden. Als erstes müssen die Pfade der Bilder von MCSE als String deklariert werden. Diese befinden sich in einem speziellen Ordner namens „03_AutomaticStartCameo“ in der Ordnerstruktur der Anwendung. Der Pfad der Datei „simulation.log“ muss ebenfalls deklariert und mit dem Namen des aktuellen Benutzers angepasst werden, da sich diese Datei normalerweise in einem Windows-Benutzerordner befindet. Als letzte Eigenschaft wird ein Timestamp aufgenommen und mit 1000 multipliziert, da MCSE eine Java-Applikation ist und die Timestamps von Java drei Werte größer sind als die von Python, diese Eigenschaft wird als „starttimestamp“ deklariert. Diese Prozedur ist dem Konstruktor des Objekts zugeordnet, und auf Wunsch des Programmierers können die Funktionen der Klasse aufgerufen werden, um MCSE-Modelle abzurufen.

Wenn die Funktion „startmodel_V2“ ausgeführt werden soll, muss zusätzlich der Pfad der „.mdzip“-Datei angegeben werden. Zuerst wird geprüft, ob die Datei mit der gewünschten Endung endet und ob dieser Pfad existiert, andernfalls wird ein „False“ zurückgegeben und eine entsprechende Fehlermeldung auf der Konsole ausgegeben. Nach der ersten Prüfung wird mit Hilfe des Moduls „Subproses“ der Pfad zum gewünschten Modell mit der Funktion „Popup“ geöffnet, da „Subproses“ wie ein Windows-Befehlsfenster funktioniert. Anschließend werden die Variablen „H“ und „W“ erstellt und mit den Pixelmaßen des Bildschirms über die Funktion „pyautogui.size“ (Höhe und Breite) belegt. Dann werden Hilfsvariablen erstellt, die erste ist ein Zähler mit dem Namen „counter“ und ein Boolean mit dem Wert „False“ und dem Namen „readytosim“, um die erste Schleife zu generieren, die den erfolgreichen Start des MCSE bestätigt. Die erste Schleife sucht mit der Funktion „pyautogui.locateOnScreen“ und den Bildschirmabmessungen in Intervallen von 1 Sekunde nach einem bestimmten Bild, in diesem Fall Abbildung 40. Wenn das Bild nicht innerhalb von 60 Sekunden gefunden wird, wird eine Fehlermeldung ausgegeben und „False“ zurückgegeben. Wenn das Bild auf dem Bildschirm gefunden wird, wird „readytosim“ auf True gesetzt und die Schleife beendet. Danach wird mit der gleichen Methode nach anderen Bildern, ein Play-Button und ein Dropdown-Button gesucht. Wenn die Bilder gefunden wird, werden deren Koordinaten in Variablen gespeichert.

Nachdem die Bilder und die Koordinaten auf dem Bildschirm gefunden wurden, beginnt die zweite Schleife der Klasse mit der Berechnung der Koordinaten der Konfigurationen. Die erste Konfiguration („1 Input“) befindet sich 0,521% der Bildschirmbreite (dieser Wert bleibt konstant) von den Koordinaten des Dropdown-Menüs entfernt. Beim ersten Durchlauf der Schleife befindet sich die erste Konfiguration 2,1% der Höhe des Bildschirms vom Dropdown-Menü entfernt, und für die folgenden Konfigurationen werden 0,55% der Höhe des Bildschirms

hinzugefügt.

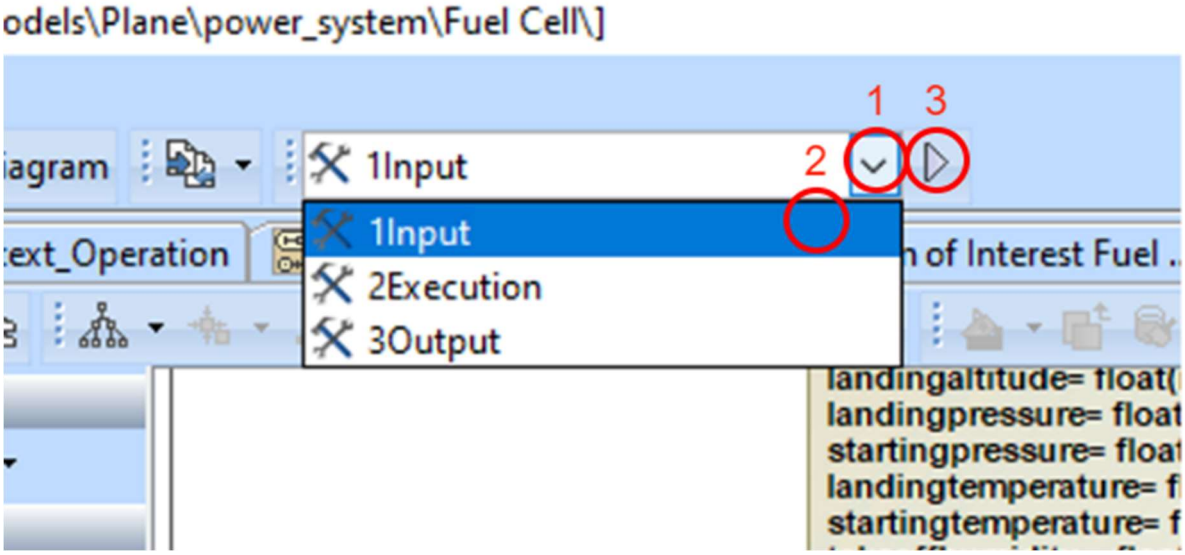


Abbildung 41: Abschnitt des MCSE Simulationsbalken.

Zum besseren Verständnis der Schleife wird die Erklärung durch Abbildung 41 ergänzt. Zuerst wird das Dropdown-Menü ausgewählt (Nummer eins in der Abbildung 41). Beim ersten Durchlauf wird die Maus um 0,521% der Bildschirmbreite nach links und um 2,1% der Bildschirmhöhe nach unten bewegt und angeklickt (Nummer 2 in der Abbildung 41). Schließlich wird der Play-Button angeklickt (Nummer 3 in der Abbildung 41). Für die nächsten Durchgänge werden nur noch 0,55% der Bildschirmhöhe zu den Koordinaten addiert, damit weitere Konfigurationen ausgewählt werden können.

Während der Playbutton betätigt wird, laufen im MCSE interne Prozesse ab, die in der Simulationslogdatei protokolliert werden. Diese werden durch die Funktionen „logchek“ und „isTheSameConfigbeeingplayed“ überwacht. Die zweite Funktion sucht nach dem Namen der Konfiguration, die beim ersten Durchlauf „1 Input“ heißen soll. Wenn sich der Name der Konfiguration im nächsten Durchlauf nicht wiederholt, beginnt die zweite Funktion nach dem Ende der Simulationsnachricht zu suchen, damit der nächste Durchlauf gestartet werden kann. Wenn sich der Konfigurationsname wiederholt, wird die Schleife beendet, die MCSE-Datei gespeichert und geschlossen.

Alle diese Schritte sind in zusammengefasster Form im Aktivitätsdiagramm in Abbildung 42 dargestellt. Die meisten Werte und Faktoren der Klasse sind experimentell bestimmt worden, jedoch mehrfach getestet, um sicherzustellen, dass die Anforderungen des Endnutzers erfüllt werden.

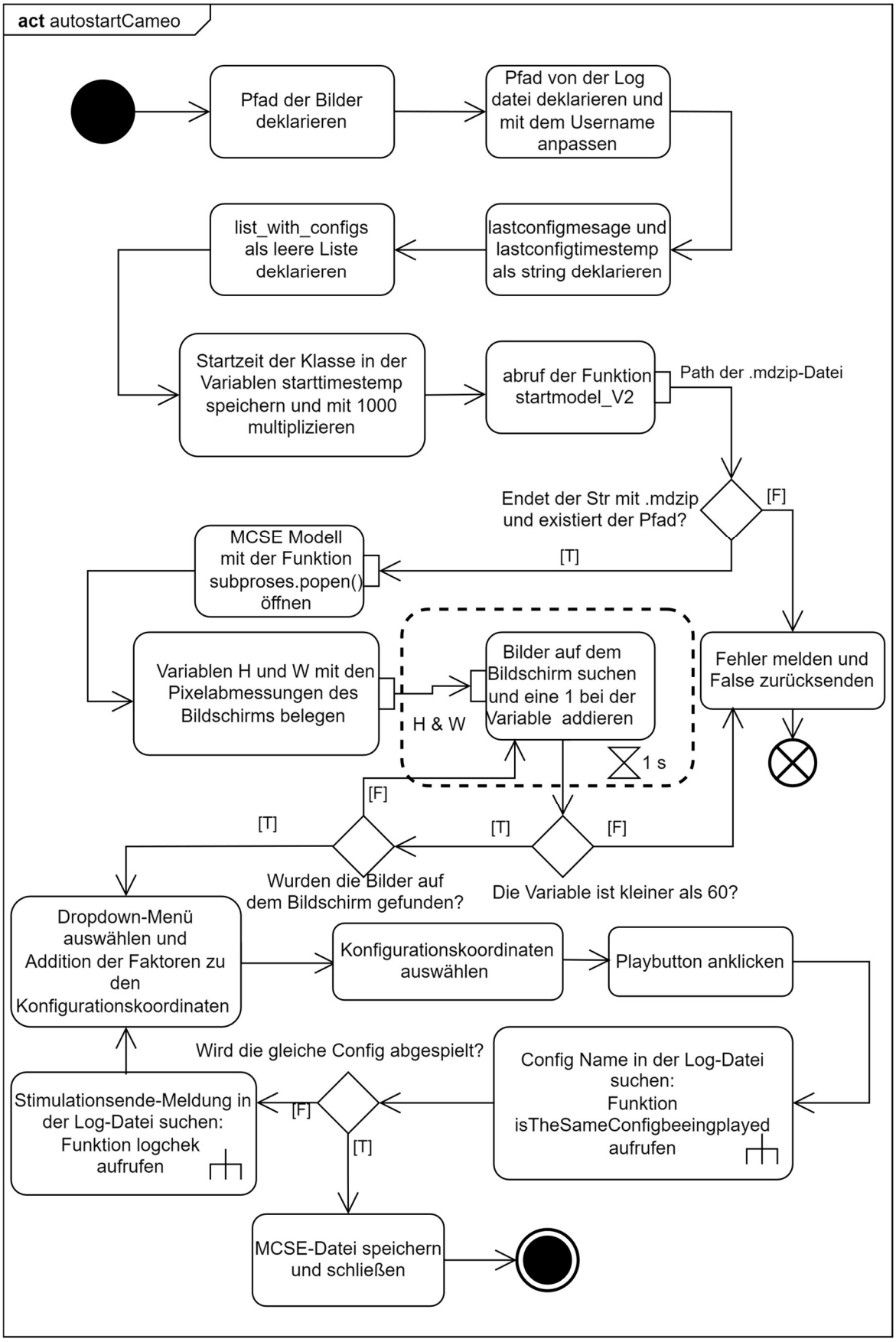


Abbildung 42: Aktivitätsdiagramm der Klasse AutostartCameo.

Da die Funktionen „logchek“ und „isTheSameConfigbeeingplayed“ nach dem gleichen Prinzip arbeiten, aber verschiednes suchen, werden sie zusammen erläutert. Es handelt sich um eine Schleife mit einer Pause von 0,5 Sekunden pro Durchlauf. Außerdem hat die Funktion einen Timer, um die Funktion zu stoppen, wenn nach einer bestimmten Zeit nichts gefunden wurde. Diese beträgt für die erste Funktion 2,5 Minuten und für die andere Funktion 15 Minuten.

Zuerst wird die Logdatei von MCSE als String ausgelesen und mit einem „Tag“ versehen, der die ganze Logdatei in die XML-Schreibweise umwandelt. Dadurch ist es möglich, die Informationen als XML-Baum zu betrachten. In diesem Format kann nach Informationen gefiltert werden. Zuerst wird nach dem Timestamp gefiltert, um nur Informationen vom Start des Programms zu bekommen. Dann wird die Informationder Konsole extrahiert, indem Nachrichten vom Typ „@logger“ und die Information „SIM_PLAIN_CONSOLE-1“ identifiziert werden. Als letztes wird nach einem „print“ in der Konsole gesucht, der den String „Cameo Config:“ im Zusammenhang weitere Informationen enthält. Die zusätzliche Information dieses „prints“ kann zum Beispiel "1Input" sein („Cameo Config: Input stared“).

Die Namen der Konfigurationen werden in einer Liste („list_with_configs“) gespeichert. Wenn sich der Name der Simulationskonfiguration wiederholt, wird die Schleife beendet, das MCSE-Modell gespeichert und geschlossen. Die zweite Funktion macht im Prinzip das gleiche mit anderen Nachrichtentypen, diese sucht nach Nachrichten des Typs „SIM_USER_OUTPUT_CONSOLE-1“ und mit der zusätzlichen Information „execution is terminated. **“ gesucht, um zu bestätigen, dass eine Simulation erfolgreich beendet wurde.

6 Ergebnisse

Um die Zuverlässigkeit der Anwendung zu überprüfen, werden verschiedene Tests durchgeführt. Dadurch wird sichergestellt, dass die Anwendung wie gewünscht ausgeführt wird.

6.1 Testszenario

Das erste Szenario ist die Simulation mehrerer einheitlicher Simulationen mit denselben Anfangswerten. Zu diesem Zweck wurde ein einzelnes MCSE-Modell, das Brennstoffzellenmodell, ausgewählt, welches zehn Mal simuliert wurde. Die Ergebnisse der generierten XML-Dateien werden verglichen und auf Fehler und/oder Inkonsistenzen untersucht werden.

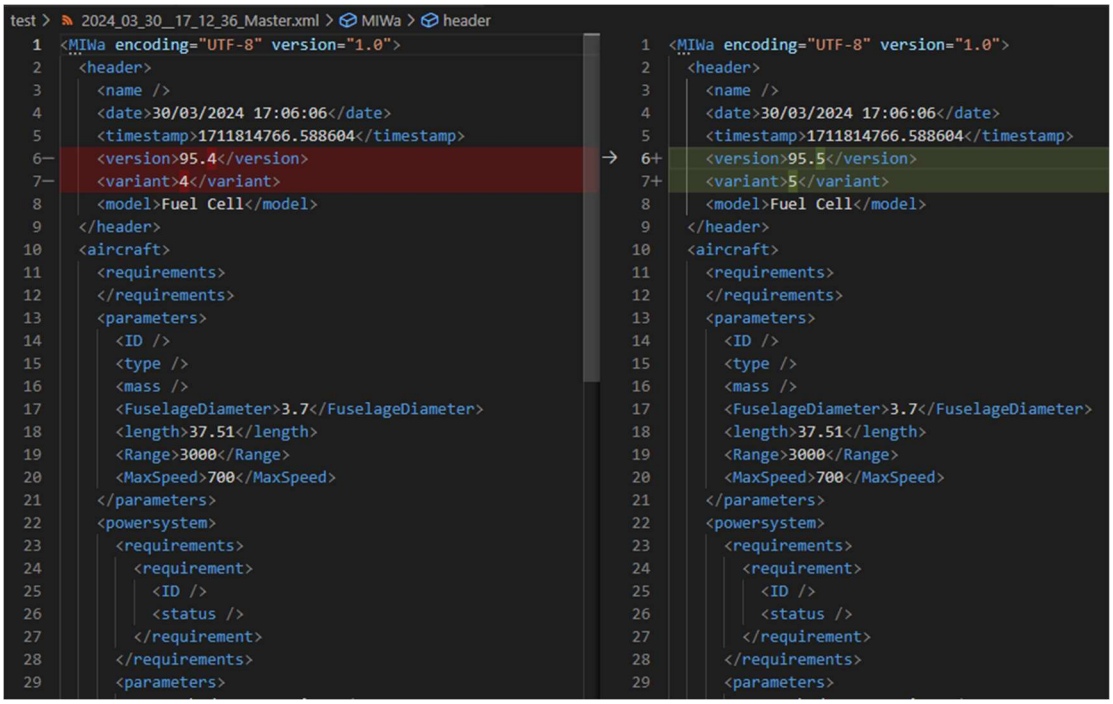


Abbildung 43: Vergleich der Ergebnisse mit Visual Studio Code

Nach den Durchläufen wurde bestätigt, dass alle Modelle ohne Probleme simuliert werden konnten, wobei jeder Durchlauf im Durchschnitt 85 Sekunden dauerte. Ziel war es, die Zuverlässigkeit beim Start mehrerer Simulationen zu testen. Mit Hilfe der Werkzeuge von Visual Studio Code war es möglich, die Datei automatisch zu vergleichen. Der Vergleich zeigt, dass sich die Dateien nur in der Version und der Variante unterschieden. Abbildung 43 zeigt einen Teil des maschinellen Vergleichs der Dateien, bei dem die Variante 4 und Variante 5 verglichen wurde. Die markierten Zeilen zeigen die Unterschiede in den Dateien. Der digitale Anhang enthält diese 10 simulierten Varianten zur Bestätigung des Vergleichs.

6.2 Testszenario

Dieser Test besteht aus einer Variante mit den folgenden Modellen: Power System, Fuel Cell, Tank, Pressure Bulkhead und Cabin. Für diesen Test wurden zwei Modelle hinzugefügt, die zu Beginn dieser Arbeit noch nicht vollständig entwickelt waren. Das erste unbekannte Modell ist das Power System, welches als übergeordnetes Modell beschrieben werden kann, da die Aufgabe dieses Modells darin besteht, die Missionsparameter vorzugeben. Zum Beispiel die Reichweite des Flugzeugs, die in diesem Test mit 3000km variiert wurde. Das zweite Modell ist das Bulkhead-Modell, das den Kabinenraum bestimmt.

Tabelle 3: Resultate des Tests 2

Modell	Art von Parameter	Wert	Einhait
Power System	StartingAltitude	0	m
Power System	Range	3000	km
Power System	TakeOffMass	73500	kg
Power System	CruiseMass	70000	kg
Power System	ThrustCruise	44685	N
Power System	TimeCruise	112	min
Fuel Cell	CruisePower	200000	W
Fuel Cell	MaxPowerFCStack	120000	W
Fuel Cell	MassHydrogen	1615	kg
Fuel Cell	Mass	108	kg
Tank	PressureTank	200000	Pa
Tank	RpInnerWall	314	Mpa
Tank	LengthTank	3.66	m
Tank	DiameterTank	3.61	m
Bulkhead	Length	4.659	m
Cabin	Cabintype	MaxPax	-
Cabin	Seats	163	-
Cabin	Lavatorys	3	-
Cabin	Galleys	1	-

Die gesamte Simulation dauerte 422 Sekunden. Der Simulationsablauf beginnt mit dem Einlesen der Startparameter, die vor der Simulation ausgewählt wurden. Diese Parameter sind in Tabelle 3 grün markiert. Nach der Berechnung werden die Ergebnisse in der XML-Datei gespeichert und sind in Tabelle 3 rot markiert. Wie z.B. die Masse an Wasserstoff, die die Brennstoffzellen benötigen.

6.3 Testszenario

Der dritte Versuch ist im Prinzip der vorherige Versuch mit einer größeren Reichweite (4000 km).

Tabelle 4: Resultate des Tests 3

Modell	Art von Parameter	Wert	Einhait
Power System	StartingAltitude	0	m
Power System	Range	4000	km
Power System	TakeOffMass	73500	kg
Power System	CruiseMass	70000	kg
Power System	ThrustCruise	44685	N
Power System	TimeCruise	149	min
Fuel Cell	CruisePower	200000	W
Fuel Cell	MaxPowerFCStack	120000	W
Fuel Cell	MassHydrogen	2131	kg
Fuel Cell	Mass	108	kg
Tank	PressureTank	200000	Pa
Tank	RpInnerWall	314	Mpa
Tank	LengthTank	4,44	m
Tank	DiameterTank	3,61	m
Bulkhead	Length	5,44	m
Cabin	Cabintype	MaxPax	-
Cabin	Seats	151	-
Cabin	Lavatorys	3	-
Cabin	Galleys	2	-

Die größere Reichweite erfordert eine größere Masse an Wasserstoff, was sich auf die Tankgröße und gleichzeitig auf die Anzahl der Sitze auswirkt. Diese Simulation dauerte 424 Sekunden.

7 Diskussion der Ergebnisse

Die in Kapitel 6 durchgeführten Tests bestätigen, dass die Hauptfunktionen der Anwendung tatsächlich zur Verfügung stehen. Für eine detaillierte Bewertung wird auf die in Tabelle 2 formulierten Anforderungen an die Anwendung verwiesen.

Tabelle 5: Angefärbte Anforderungstabelle mit Priorisierung

	Anfoderung	Kategorie
1	Automatisch MCSE Modelle mit den entsprechenden Konfigurationen ausführen zu können.	must
2	Auswahl der auszuführenden Modelle.	must
3	Generierung einer XML-Datei für den Start des Simulationslaufs	must
4	Einführung eines Zeitstempels und einer Versionsnummer in der XML-Anfangsdatei	must
5	Einführung der Namen der Simulationsmodelle in der XML-Datei	must
6	Verschieben der XML-Datei in das gewünschte Verzeichnis.	must
7	Kommunikation der Prozesse über die GUI	must
8	Positionierung der Komponenten über die GUI	must
9	Auswahl einer beliebigen Anzahl von Simulationen	must
10	Parametervariationen ermöglichen	must
11	Freitext-Eingabemöglichkeiten für Eingabeparameter anbieten	must
12	Anbindung externer Datenbanken für die Startparameter der Modelle	must
13	Mitteilung von Fehlermeldungen über die GUI	should
14	Auswahl zwischen Standard- und Expertenmodus	should
15	Einfügen eines Play-Buttons zum Starten der Simulationen	should
16	3D-Visualisierung der Varianten	should
17	Intuitive und benutzerfreundliche Gestaltung	should
18	Visualisierung der ausgewählten Modelle	should
19	Einfache Erweiterung und Pflege von Modellen und Funktionalitäten	should
20	Reproduzierbare und einfache Ordnerstruktur zur Erweiterung der Modelle	should
21	Möglichst neutrale und erweiterbare Programmierung zur allgemeinen Nutzung und Wiederve	should
22	Erstellung einer Log-Datei für alle generierten Meldungen	should

Tabelle 5 enthält dieselben Informationen wie Tabelle 2, ist jedoch farblich gekennzeichnet, wenn die Anforderung erfüllt (grün), nicht erfüllt (rot) oder noch nicht bestätigt (grau) ist.

Die erste nicht erfüllte Anforderung ist die Visualisierung der Ergebnisse nach Abschluss der Simulation. Dies sprengte den Rahmen der Bachelorarbeit, da erst ein automatischer Catia-Lizenzüberprüfer und -Lizenzaktivator programmiert werden müsste, damit Catia mit den Floating Lizenzen der Centerline Design GmbH zuverlässig funktioniert. Aus diesem Grund und der geringen Catia-Expertise des Entwicklers war die Anbindung von Catia in der GUI zeitlich nicht realisierbar.

Die Positionierung von Komponenten durch die GUI war bis jetzt nicht erreichbar, da keine der MIWa-Modelle diese Möglichkeit in den Anfangsparameter anbietet. Allerdings kann dies durch die einfache Erweiterung der Anfangsparameter erweitert werden.

Die grau unterlegten Anforderungen sind aus Sicht des Entwicklers schwer zu bestätigen, da dazu eine externe Testphase gestartet werden muss. Durch den Einsatz von externen Anwendern und die Erweiterung durch externe Programmierer kann diese Anfoderung beantwortet werden. Jedoch ist zumindest bestätigt, dass durch den Einsatz von OOP die einzelnen Klassen der Anwendung gekapselt werden können, was die Übersichtlichkeit des Programms verbessert. Dadurch können die einzelnen Klassen erweitert, vererbt oder ersetzt werden.

Die erste Anforderung wurde durch die Klasse „autostartcameo“ aus Kapitel 5.12 erfüllt, jedoch kann diese Klasse noch erweitert werden, um z.B. die Anforderungen von magic cyber systems engineer zu überprüfen. In Kapitel 5.12 wurde erwähnt, dass es zusätzliche Möglichkeiten zur externen Simulation der Modelle gibt. Jedoch erfüllte diese Möglichkeit nicht die Anforderungen der MIWa Systementwickler, da die Ergebnisse nicht gespeichert werden konnten. Es ist möglich, dass in zukünftigen Versionen ein „Application Programming Interface“ (API) von MCSE zur Verfügung gestellt wird, um die Ausführung zuverlässiger zu machen, was die Geschwindigkeit der entwickelten Anwendung deutlich erhöhen könnte.

Die Klasse „SimulationFrame“ von Kapitel 5.11 erfüllt die Anforderungen 2 und 19 durch die Bereitstellung von drei Schaltflächen. Die zwei Schaltflächen (Plus und Minus) erzeugen bzw. löschen Simulationen, da diese in direktem Zusammenhang mit der Klasse "Notebook Simulation Frames" aus Kapitel 5.7 stehen. Eine Verbesserung für diese Klassen Wird die Vergabe eines individuellen Speichernames für die Simulation sein. Die erste Schaltfläche startet die Simulation und ist mit einem Play-Symbol gekennzeichnet. Wird der Button gedrückt, wird automatisch eine XML-Datei erzeugt, die den Zeitstempel der Simulation mit einer Versionsnummer und der Modellreihe enthält. Dies entspricht den Anforderungen 3, 5 und 13. Nach Beendigung der Simulation wird abgefragt, ob die Simulation an einem anderen Ort gespeichert werden soll (Punkt 21).

Zum besseren Verständnis und zur Verwaltung der Simulationen sollte die Versionsnummer nicht von der Anzahl der durchgeführten Simulationen abhängen, sondern von den Simulationsmodellen. Das heißt, wenn eine Simulationsreihe aus „Power System“, „Fuel Cell“, „Tank“ und “Cabin“ besteht, haben alle Simulationen, die aus dieser Reihe bestehen, die gleiche Versionsnummer, z.B. 1, und unterscheiden sich dann von der Variante. Diese Änderung wird in zukünftigen Versionen der Anwendung implementiert.

Die Kommunikation der Prozesse ist ein wichtiger Teil der Anwendung und wurde in den Anforderungen 4, 14 und 17 aufgenommen. Die Lösungen zu diesen Anforderungen sind in den Kapiteln 5.2 und 5.3 beschrieben. Zusammengefasst gibt es zwei Kommunikationskanäle, über jede Funktion in der gesamten Anwendung Meldungen ausgeben kann. Diese Meldungen werden nicht nur in grafischen Oberflächen angezeigt, sondern auch in einer Log-Datei gespeichert, um z.B. Fehler zu dokumentieren.

Dank der Ordnerstruktur erkennt die Anwendung teilweise, wo sich bestimmte (Modelle, Master.xml) Dateien befinden, was nach Anforderung 10 erfüllt sein muss. Die Deklaration von Modellreihen und neuen Parametern setzt jedoch voraus, dass der Administrator mit XML-

Dateien vertraut ist. Dies kann zu menschlichen Fehlern führen. Dieses Problem wird durch die Verwendung von Excel-Tabellen gelöst, für die Kenntnisse im Umgang mit Excel-Tabellen erforderlich sind. Aufgrund des Zeitmangels und der nicht abgeschlossenen Testphase des „Exel2XML-Übersetzers“ wurde in dieser Arbeit nicht näher darauf eingegangen.

Wenn eine Reihe von Modellen ausgewählt wird, werden verschiedene Notizbücher erstellt, in denen die spezifischen Parameter der Modelle personalisiert werden. Diese Aktion erfüllt Anforderung 8. Der Benutzer hat die Möglichkeit, am Anfang der Reihe von Parametern zwischen einem Experten- und einem Standardbenutzer zu wählen. Die erste Variante erlaubt dem Benutzer Werte in die Eingabefelder zu schreiben, die zweite Möglichkeit erlaubt nur die Auswahl zwischen vordefinierten Werten, dadurch sind die Anforderungen 6, 12 und 15 erfüllt. In Zukunft ist geplant, dass die Eingabefelder zwischen Buchstaben und Zahlen unterscheiden können, um Fehler zu vermeiden.

8 Zusammenfassung Ausblick

Zusammenfassend wurde in dieser Arbeit eine Anwendung entwickelt, die MCSE-Dateien teilweise automatisch ausführt. Damit können alle Konfigurationen der verschiedenen Modelle der Flugzeugsysteme schneller und effizienter generieret werden, indem der Benutzer die Startparameter in der Benutzeroberfläche eingibt und die Simulation der Modelle automatisch abläuft. Zusätzlich besteht die Möglichkeit, die Startparameter der Modelle durch die Eingabe von Daten oder durch den Verweis auf eine externe Datenbank zu erweitern. Am Ende der Simulation wird die simulierte XML-Datei in einem Verzeichnis gespeichert und bei Bedarf an einen anderen Ort exportiert. Durch die angewandte Ordnerstruktur können jederzeit neue Modelle, Parameter und Anwendungen hinzugefügt werden, indem die Modelle einfach in den gewünschten Ordner importiert und deklariert werden. Ein Vorteil der Anwendung ist die Planung mehrerer Simulationen, so dass sie später nacheinander ohne menschliche Aufsicht simuliert werden können. Allerdings darf der Computer während der Simulation nicht benutzt werden, da die Simulation mit der Maus durchgeführt wird.

Nicht alle Anforderungen, die zu Beginn des Projekts (siehe Tabelle 2) definiert wurden, konnten erfüllt werden (siehe Kapitel 7). Eine dieser Anforderungen war die 3D-Visualisierung der Ergebnisse mit Hilfe von CAD-Tools. Aus Zeitgründen konnte diese Integration nicht realisiert werden. Python verfügt jedoch über die Werkzeuge, um eine stabile Anbindung an CAD-Programme zu realisieren und diese zu steuern. Eine weitere Anforderung, die nicht erfüllt werden konnte, war die Positionierung von Bauteilen in der GUI, aber da keines der bisher eingeführten MCSE-Modelle diese Möglichkeit bietet, könnte diese Möglichkeit nicht angeboten werden. Aufgrund der einfachen Erweiterbarkeit der Anwendung können aber immer wieder und ohne Einschränkung neue Startparameter mit Default-Optionen gespeichert werden.

Es handelt sich um eine Betaversion einer Anwendung, die noch um neue Funktionalitäten und Tests erweitert werden muss, wenn diese Software tatsächlich eingesetzt werden soll. Dennoch hat sich die Software in den Testszenarien als nützlich erwiesen, da die Integration in andere Projekte oder Parameter und Modelle einfach gestaltet werden kann.

Die Verwendung von Shortcuts im MCSE, die von der Anwendung über Python-Skripte gesteuert werden, bietet einen deutlichen Mehrwert gegenüber den beschriebenen Klickbefehlen. Aufgrund interner Fehler im MCSE kann diese Option jedoch nicht genutzt werden. Eine andere Möglichkeit besteht darin, die Modellierungssoftware so zu erweitern, dass die Anwendung den Dateityp erkennt und die Simulation unabhängig vom verwendeten Werkzeug startet, z. B. die Ausführung von Matlab- oder Papyrus-Modellen.

Ein weiterer Punkt, der in zukünftigen Versionen eingeführt werden könnte, ist die Verwendung eines neuen Dateityps, in dem die Informationen der aktuellen Simulation gespeichert und simuliert werden können. Dies hätte den Vorteil, dass die Anwender zwischen

den Parametern von anderen wechseln könnten, so dass die Ergebnisse von anderen schneller reproduziert werden könnten.

Alle in der Aufgabenstellung in Kapitel 1.3 aufgeführten Punkte wurden bearbeitet. Die Ergebnisse zeigen, dass die Anwendung die meisten Anforderungen erfüllt, so dass die Benutzer Vergleiche zwischen Modellen und Parametern effizient durchführen können.

Literaturverzeichnis

[1] 9. Classes (o. D.): Python Documentation, [online] https://docs.python.org/3/tutorial/classes.html Abruf am 15.01.2024.

[2] Bachmann, Bernhard (1990): Grosses Lexikon der Computerfachbegriffe, ISBN: 9783883222578.

[3] ChatGPT (o. D.): [online] https://chat.openai.com/c/8ffa317e-9957-4ed2-9564-771bbd701a3c.

[4] Crabtree, G. W./M. S. Dresselhaus (2008): The Hydrogen Fuel Alternative, in: *Mrs Bulletin*, Bd. 33, Nr. 4, S. 421–428, [online] doi:10.1557/mrs2008.84 Abruf am 12.01.2024.

[5] difflib — Helpers for computing deltas (o. D.): Python Documentation, [online] https://docs.python.org/3/library/difflib.html Abruf am 14.01.2024.

[6] Engmann, Klaus (2009): *Technologie des Flugzeuges*, ISBAN: 9783834331595.

[7] Extensible Markup Language (XML) 1.0 (Fifth Edition) (o. D.): [online] https://www.w3.org/TR/REC-xml/ Abruf am 20.01.2024.

[8] Fuchs, Mara et al. (2023): Modellierungsstrategie zur Analyse des Einflusses von alternativen Antriebskonzepten auf Kabinensysteme, in: *ResearchGate*, [online] https://www.researchgate.net/publication/375926796_Modellierungsstrategie_zur_Analyse_des_Einflusses_von_alternativen_Antriebskonzepten_auf_Kabinensysteme , Abruf am 10.02.2024.

[9] General Python FAQ (o. D.): Python Documentation, [online] https://docs.python.org/3/faq/general.html#what-is-python Abruf am 18.01.2024.

[10] Green Aviation Technologies (2022): hamburg.de, [online] https://www.hamburg.de/bwi/medien/16634612/2022-11-02-bwi-green-aviation-technologies/ , Abruf am 16.01.2024.

[11] Gualtieri, Maurizio et al. (2022): Emission Factors of CO2 and Airborne Pollutants and Toxicological Potency of Biofuels for Airplane Transport: A Preliminary Assessment, in: *Toxics*, Bd. 10, Nr. 10, S. 617, [online] doi:10.3390/toxics10100617, Abruf am 4.02.2024.

[12] Incose (2023): *INCOSE Systems Engineering Handbook*, John Wiley & Sons, ISBN: 9781119814290.

[13] *Internationales elektrotechnisches Wörterbuch* (2014): DKE/UK 931.2, [Norm] https://www.dke.de/de/normen-standards/dokument?id=7045967&type=dke%7Cdokument .

[14] json — JSON encoder and decoder (o. D.): Python Documentation, [online] https://docs.python.org/3/library/json.html Abruf am 21.01.2024.

[15] Kaiser, Richard (2021): C++ mit Visual Studio 2019, Springer-Verlag, ISBN: 9783662594759.

[16] Lahres, Bernhard et al. (2018): *Objektorientierte Programmierung: Das umfassende Handbuch. Lernen Sie die Prinzipien guter Objektorientierung,* ISBN: 9783836262477.

[17] Larminie, James/Andrew Dicks (2003): *Fuel cell systems explained*, Wiley-Blackwell, ISBN: 9783836262477.

[18] Magic Model Analyst Documentation (o. D.): No Magic Product Documentation, [online] https://docs.nomagic.com/display/MSI2021xR2/Magic+Model+Analyst+Documentation, Abruf am 17.02.2024.

[19] No Magic - Dassault Systèmes (2024): Dassault Systèmes, [online] https://www.3ds.com/products/catia/no-magic, Abruf am 17.02.2024.

[20] Nyamsi, Eric A. (2020): *IT-Lösungen auf Basis von SysML und UML: Anwendungsentwicklung mit Eclipse UML Designer und Eclipse Papyrus*, Springer-Verlag, ISBN: 9783658290573.

[21] os — Miscellaneous operating system interfaces (o. D.): Python Documentation, [online] https://docs.python.org/3/library/os.html Abruf am 22.01.2024.

[22] pandas (2024): PyPI, [online] https://pypi.org/project/pandas/ Abruf am 15.01.2024.

[23] pillow (2024): PyPI, [online] https://pypi.org/project/pillow/ Abruf am 15.01.2024.

[24] Rothfuss, Gunther/Christian Ried (2001): *Content Management mit XML*, Springer-Verlag New York Incorporated.

[25] Saumweber, Walter (2015): *Windows 10 - Das große Handbuch: Einstieg, Praxis, Profi-Tipps - das Kompendium zu Windows 10. Der Klassiker für die Anwender-Praxis*, ISBN: 9783842101623.

[26] time — Time access and conversions (o. D.): Python Documentation, [online] https://docs.python.org/3/library/time.html Abruf am 16.01.2024.

[27] tkinter — Python interface to Tcl/Tk (o. D.): Python Documentation, [online] https://docs.python.org/3/library/tkinter.html Abruf am 18.01.2024.

[28] Töpler, Johannes/Jochen Lehmann (2017): *Wasserstoff und Brennstoffzelle: Technologien und Marktperspektiven*, Springer-Verlag, ISBN: 9783662533604.

[29] Tschirner, Christian (2023): Was ist MBSE? - Two Pillars, Two Pillars GmbH, [online] https://www.two-pillars.de/was-ist-mbse/ Abruf am 27.02.2024.

[30] ttkbootstrap - ttkbootstrap (o. D.): [online] https://ttkbootstrap.readthedocs.io/en/latest/ .

[31] Uppal, Rajesh (o. D.): Model-Based Systems Engineering (MBSE) for Satellites and Aerospace - International Defense Security & Technology, International Defense Security & Technology, [online] https://idstch.com/technology/ict/model-based-systems-engineering-mbse-for-satellites-and-aerospace/, Abruf am 11.01.2024.

[32] Using model-based systems engineering to develop the next-generation A350 XWB (o. D.): Siemens Resource Center, [online] https://resources.sw.siemens.com/en-US/case-study-airbus-a350-xwb Abruf am 11.01.2024.

[33] Von Joerg, Garrel/Carlos Jahn (2022): Design Framework for the Implementation of AI-based (Service) Business Models for Small and Medium-sized Manufacturing Enterprises, in: *Journal Of The Knowledge Economy*, Bd. 14, Nr. 3, S. 3551–3569, [online] doi:10.1007/s13132-022-01003-z Abruf am02.02.2024.

[34] Weigend, Michael (2013): *Python 3: Lernen und professionell anwenden*, mitp., ISBN: 9783826694561.

[35] Weilkiens, Tim (2014): *Systems Engineering mit SYSML/UML: Anforderungen, Analyse, Architektur (Mit einem Geleitwort von Richard Mark Soley)*, ISBN: 9783864900914.

[36] Welcome to PyAutoGUI's documentation! — PyAutoGUI documentation (o. D.): [online] https://pyautogui.readthedocs.io/en/latest/ Abruf am 05.10.2023.

[37] Wessel, Ivo (1998): *GUI-Design: Richtlinien zur Gestaltung ergonomischer Windows-Applikationen*, ISBN: 9783446219618.

[38] What Is MATLAB? (o. D.): MATLAB & Simulink, [online] https://de.mathworks.com/discovery/what-is-matlab.html Abruf am 17.01.2024.

[39] xml.etree.ElementTree — The ElementTree XML API (o. D.): Python Documentation, [online] https://docs.python.org/3/library/xml.etree.elementtree.html Abruf am 16.01.2024.

Erklärung der selbständigen Erstellung des Berichts

Hiermit versichere ich, dass ich den vorliegenden Bericht ohne fremde Hilfe und nur mit den angegebenen Hilfsmitteln erstellt habe. Des Weiteren sind alle wörtlich oder dem Sinn nach aus anderen Werken entnommenen Stellen unter Angabe der Quelle kenntlich gemacht worden.

Hamburg, den 26.04.2024

Francisco Hoyos Garcia

Anhang

Kurze Anleitung

GUI MIWa

Automatisierte Steuerung von Cameo-Modellen

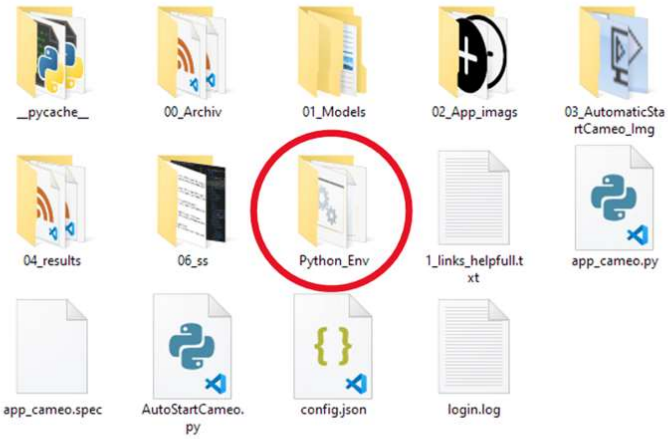
Anleitung
Workspace Vorbereitung
Nutzung
Erweiterung

21.04.2024 2 ENTERLINE DESIGN

Workspace Vorbereitung

Anforderungen

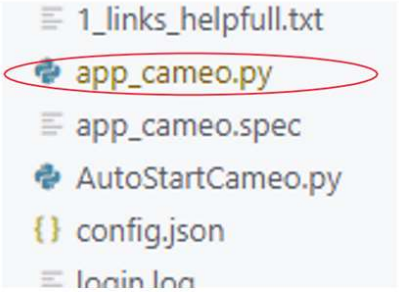
- Python-Grundlagen: Im Ordner der Abgabe befindet sich ein Python Environment mit allen für das Projekt relevanten Modulen.
- Cameo Systems Modeler installieren
- Matlab installieren



21.04.2024 3 ENTERLINE DESIGN

Python Grundlagen

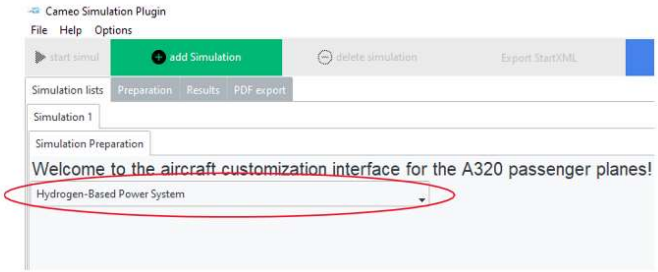
- Wenn die Python-Umgebung deklariert wurde, ist es notwendig, den Ordner der Abgabe, Workspace, im gewünschten Code-Editor zu öffnen.
- Damit die GUI ausgeführt werden kann, muss die Datei App_cameo.py ausgeführt werden.



Nutzung

Simulation Vorbereitung

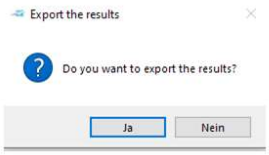
- Beim Ausführen der Anwendung wird das Hauptfenster generiert, in dem bereits die Simulationsmöglichkeit angeboten wird.



Nutzung

Ergebnisse der Simulation exportieren

- Am Ende der Simulation erscheint ein Fenster mit der Frage, ob die Simulationsergebnisse exportiert werden sollen. Falls gewünscht, können die Ergebnisse in einem Ordner gespeichert werden.

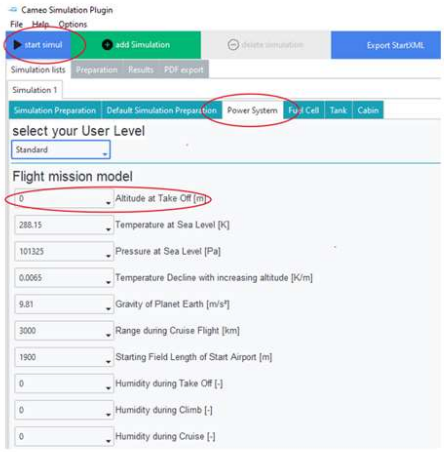


Nutzung

Simulation Vorbereitung

- Wenn eine Simulation ausgewählt wurde, werden weitere interne Fenster generiert, in denen die Startparameter für jede Simulation angepasst werden können. Falls erforderlich, kann die Benutzerebene angepasst werden, um die Startparameter selbst einzugeben.

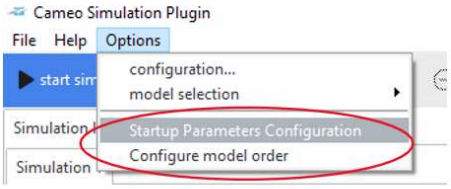
- Wenn alle Parameter festgelegt sind, kann die Simulation mit dem Play-Button gestartet werden. Es ist zu beachten, dass während der Simulation keine weitere Bedienung des Rechners möglich ist.



Erweiterung

Parameter und Modellreihenfolge

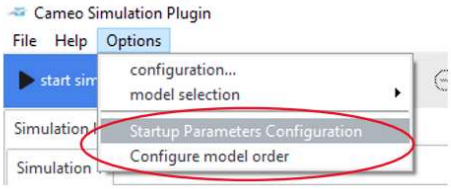
- Das Hinzufügen neuer Parameter oder Modelle ist möglich, indem die Werte, Parameter oder Modelle in der Menüleiste deklariert werden.



Erweiterung

Parameter und Modelle hinzufügen

- Das Hinzufügen neuer Parameter oder Modelle ist möglich, indem die Werte, Parameter oder Modelle in der Menüleiste deklariert werden.



Erweiterung

Parameter hinzufügen

- Wenn die Option „Startup Parameter Configuration“ ausgewählt wird, öffnet sich eine Excel-Tabelle, in der die Parameter nach Modellen geordnet sind. Wenn ein neuer Wert hinzugefügt werden soll, müssen zusätzliche Informationen wie Name, Tag im XML, 3 Standardwerte, Einheit und eine Beschreibung des Parameters angegeben werden.
- Die Exeltabelle muss gespeichert und geschlossen werden, danach muss die Anwendung erneut geöffnet werden, damit die Werte erfolgreich aktualisiert werden.

System	System Description	Parameter Group	Parameter Name	Parameter Description
Power System	Flight mission model		CruiseStartAltitude	Altitude at the start of cruise
Power System	Flight mission model		CruiseFinalAltitude	Altitude at the end of cruise
Power System	Flight mission model		DescentStartAltitude	Altitude at the start of descent
Power System	Flight mission model		DescentFinalAltitude	Altitude at the end of descent
Power System	Flight mission model		ThrustTakeOff	Max. Thrust during TakeOff
Power System	Flight mission model		ThrustClimb	Average Thrust during Climb

Erweiterung

Modellreihenfolge hinzufügen

- Um die Modellreihenfolge zu erweitern, muss die Option "Configure model order" ausgewählt werden, was wiederum eine Excel-Tabelle öffnet, die den Namen der Modellreihenfolge, die Beschreibung und die verschiedenen Pfade, die zum Modell führen, enthält.

name	description	model	model
Hydrogen-Based Power System		/power_system/Power System	/power_system/Hydrogen Engine
Standard Kerosene Engine with Fuel Cells		/power_system/Power System	/power_system/Fuel Cell
Standard Kerosene Engine with Fuel Cells and distribution system		/power_system/Power System	/power_system/Fuel Cell
Electric Engine		/power_system/Electric Engine	/power_system/Fuel Cell
just FC	just one Model	/power_system/Fuel Cell	